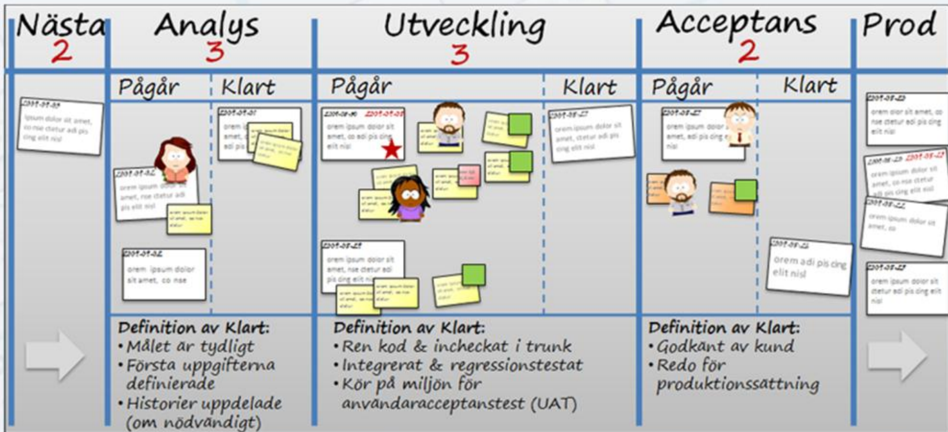


Kanban och Scrum
få det bästa av två världar

Henrik Kniberg & Mattias Skarin

Förord av Mary Poppendieck och David Anderson

ENTERPRISE SOFTWARE
DEVELOPMENT SERIES

InfoQ queue

Kanban och Scrum – få det bästa av två världar

Henrik Kniberg & Mattias Skarin

Förord av Mary Poppendieck & David Anderson

Översättning från engelskt original av
Johan Natt och Dag, Riada AB

C4Media, utgivare av InfoQ.com.

Den här boken är en del av bokserien InfoQ Enterprise Software Development.

För mer information eller beställning av denna bok, vänligen kontakta books@c4media.com (på engelska).

Redaktionschef: Diana Plesa
Omslag: Bistrian IOSIP
Sammansättning: Accurance

CIP Cataloguing in Publication:
ISBN: 978-0-557-13832-6

Översättning av Johan Natt och Dag, riada, www.riada.se.

Innehåll

Ordlista

Trots att många är vana att använda engelska uttryck så kan de ibland upplevas som ett hinder för att verkligen förstå. Nedanstående svenska begrepp har översättaren använt i boken för engelska uttryck som också är vanliga att använda i svenska.

Engelska	**Svenska**
Agile	Lättrörlig, Agil
Backlog	Behovslista, uppgiftslista
Batch	Sats
Burndown chart	Progressdiagram
Cadence	Rytm / Taktslag
Commit (oneself)	Förbinda (sig), åta sig
Daily Scrum	Dagligt Scrummöte
Daily standup (meeting)	Dagligt ståuppmöte
Deliver	Leverera
Deploy	Driftsätta
Development team	Utvecklingslag
Epic	Epos
Feedback loop	Återkopplingsloop
First-line support	First-line support
Item (backlog)	Uppgift (från behovs/uppgiftslistan)
Iteration	Iteration
Lean	Slimmad, Lean
One-piece flow	Enuppgiftsflöde
Potentially shippable code	potentiellt leverbar kod
Product backlog	Behovslista för produkt
Pull (work)	Dra (arbete)
Pull system	Pull-system (”sugsystem”)
Release	Leverera / leverans
Retrospective	Återblick
Scope	Omfattning
Scrum of Scrums	”Scrum of scrums”

SLA	Överenskommen servicenivå
Sprint backlog	Uppgiftslista för sprint
Stop the line	Stoppa bandet
Story point, sp	Användarberättelsepoäng, abp
Support team	Supportlag
Swimlane	Simbana
Task	Aktivitet
Team	Lag
Team member	Lagspelare
Time-boxed	Tidsbegränsad
User story	Användarberättelse
WIP (Work in Progress)	PA (Pågående Arbete)

Scrum-termerna är delvis hämtade från Krister Kauppis ”Scrum på svenska”.

Förord av Mary Poppendieck

Henrik Kniberg tillhör den sällsynta skaran som kan extrahera essensen i en komplicerad situation, separera kärnidéerna från tillfälliga distraktioner och ge en kristallklar förklaring som är otroligt lätt att förstå. I den här boken gör Henrik ett lysande arbete med att förklara skillnaden mellan Scrum och Kanban. Han tydliggör att de här metoderna endast är verktyg och att det du egentligen vill ha är en komplett verktygslåda med verktyg vars användning, styrkor och begräsningar du verkligen förstår.

I den här boken får du reda på vad Kanban handlar om, dess styrkor och begränsningar och när det används. Du får också en bra lektion i hur och när du kan bli bättre i Scrum – eller i något av de andra verktyg som du för tillfället använder. Henrik klargör att det viktigaste inte är vilket eller vilka verktyg du börjar med utan hur du ständigt förbättrar din användning av dem och att du hela tiden fyller på din verktygslåda.

Den andra delen, av Mattias Skarin, gör boken ännu mer kraftfull genom att guida dig genom en verklig situation där Scrum och Kanban tillämpas. Här får du exempel på hur verktygen använts, både var för sig och i kombination, för att förbättra utvecklingsprocesser. Du kommer att upptäcka att det inte finns ett ”bästa” sätt att göra saker. Du måste nämligen tänka efter själv och, baserat på den egna situationen, själv räkna ut vad som är nästa steg mot ett bättre sätt att utveckla programvara.

Mary Poppendieck

Förord av David Anderson

Kanban bygger på en mycket enkel idé. Pågående Arbete (PA) bör begränsas och något nytt bör påbörjas först när befintligt arbete levereras eller *dras* av en efterföljande funktion i arbetsprocessen. Ett kanban (eller signalkort) innebär att en visuell signal skapas för att indikera att nytt arbete kan dras eftersom pågående arbete inte motsvarar den överrenskomna omfattningen. Detta låter varken särskilt revolutionerande eller som att det i grunden skulle påverka prestation, kultur, förmåga och mognad i ett team och dess omgivande organisation. Det som är häpnadsväckande är att det gör det! Kanban verkar vara en sådan liten förändring och ändå förändrar det allt i en verksamhet.

Det vi har insett är att Kanban är en metod för förändringsarbete. Det är inte en process eller livscykel för programvaruutveckling eller projektledning. Kanban är ett sätt att införa förändring i en befintlig programvaruutvecklingsprocess eller projektledningsmetodik. Principen för Kanban är att du börjar med hur du gör nu. Du får förståelse för din nuvarande process genom att kartlägga värdeflödet och följer sedan gränserna för PA i varje steg i den processen. Sedan låter du arbete flöda genom systemet genom att dra arbete när Kanban-signaler genereras.

Kanban har visat sig användbart för team som arbetar med agil programvaruutveckling men det börjar också bli intressant för team som har ett mer traditionellt tillvägagångssätt. Kanban införs som en del av ett Lean-initiativ för att förvandla kulturen i en organisation och uppmuntra ständiga förbättringar.

Eftersom PA är begränsat i ett Kanban-system, tenderar allt som av någon anledning blir blockerat att strama upp systemet. Om en viss mängd arbetsuppgifter blockeras stannar hela processen upp. Det får effekten att hela teamet och den större organisationen fokuseras på att lösa problemet, häva blockeringarna och återställa flödet.

Kanban använder en visuell kontrollmekanism för att följa arbete som flödar genom de olika stadierna i värdeströmmen. Vanligtvis används en vit tavla med klisterlappar eller ett elektroniskt kortsystem för väggen. Bäst är nog att använda både och. Den öppenhet som detta genererar bidrar också till en kulturell förändring. Agila metoder har bidragit till att ge insikt i PA, avslutat arbete och mätningar som till exempel hastighet (mängden arbete som utförs i en iteration). Kanban går dock ett steg längre och ger insyn i processen och dess flöde. Kanban exponerar flaskhalsar, köer, föränderlighet och slöseri som alla påverkar organisationens resultat i termer av mängd värdefullt levererat arbete och tiden för att leverera det. Kanban ger gruppmedlemmar och externa intressenter insyn i effekten av sina handlingar (eller overksamhet). Tidiga fallstudier visar att Kanban, som sådan, förändrar beteenden och uppmuntrar till ökad samverkan på arbetsplatsen. Insynen i och påverkan på flaskhalsar, slöseri och föränderlighet uppmuntrar också till diskussion om förbättringar och team börjar snabbt genomföra förbättringar av sin process.

Som ett resultat av detta uppmuntrar Kanban inkrementell utveckling av befintliga processer och ett förändringsarbete som generellt är i linje med Lean och agila värden. Kanban kräver inte en radikal revolution av hur människor fungerar. Snarare uppmuntras gradvis förändring. Det är förändring som förstås och godkänns i samförstånd mellan utförarna och deras medarbetare.

Genom sitt pull-system uppmuntrar Kanban också fördröjt åtagande både vad gäller prioritering av nytt arbete och leverans av utfört arbete. Typiskt kommer teamen överens om en rytm för prioriteringsmöten med intressenterna då det bestäms vad som ska göras härnäst. Dessa möten kan hållas ofta eftersom de vanligtvis är mycket korta. En mycket enkel fråga behöver besvaras, t.ex. ”sedan vårt förra möte har två luckor blivit lediga. Vår nuvarande cykeltid är 6 veckor till leverans. Vilka två saker vill ni helst levereras om 6 veckor?”. Detta har en dubbel inverkan: en enkel fråga brukar få ett snabbt och högkvalitativt svar och det gör mötet kort. Frågans beskaffenhet betyder att arbetsåtagandet fördröjs till sista tillfället att bestämma sig. Detta ökar rörligheten genom att hantera förväntningar,

korta cykeltider från åtagande till leverans och eliminera omarbetningar eftersom risken att prioriteringarna förändras minimeras.

Ett sista ord om Kanban är att effekterna av att begränsa PA är att cykeltider blir förutsägbara och leveranser mer tillförlitliga. ”Stoppa bandet”-principen för att hantera hinder och buggar verkar också främja mycket hög kvalitet och kraftigt minska omarbetningar.

Emedan allt detta kommer att bli uppenbart med hjälp av de underbart tydliga förklaringarna i denna bok så kommer det förbli otydligt hur vi nått hit. Kanban blev inte uttänkt på en eftermiddag genom någon fantastisk uppenbarelse. Det uppstod snarare successivt under flera år. Många av de djupa psykologiska och sociologiska effekter som förändrar kultur, förmåga och mognad hos organisationer kunde man aldrig föreställa sig. Snarare blev de upptäckta. Flera resultat från Kanban är icke-intuitiva. Vad som verkar vara ett mycket mekaniskt synsätt – begränsa PA och dra arbete – visar sig ha djupgående effekter på människor och hur de samverkar och samarbetar med varandra. Varken jag eller någon annan som arbetade med Kanban i dess vagga förväntade sig detta.

Jag utövade det som blev Kanban som ett förhållningssätt till förändring som skulle möta minimalt motstånd. Detta stod klart för mig redan 2003. Jag eftersträvade också de mekaniska fördelarna. När jag vid den tiden, genom tillämpning av Lean-metoder, upptäckte att om hanteringen av AP verkade vettig så skulle begränsningen av det var vettigare: det eliminerade tiden att hantera det. 2004 bestämde jag mig därför för att försöka genomföra ett pull-system från grunden. Jag gavs tillfälle när en chef på Microsoft bad mig hjälpa honom att hantera förändringar i hans team som arbetade med underhåll och uppgraderingar av interna IT-applikationer. Det första genomförandet var baserat på restriktionsteorins pull-system, kallat Drum-Buffer-Rope. Det var en stor framgång: cykeltiden minskade med 92%, genomströmningen ökade med mer än 3 gånger och predikterbarheten (förfallodagsprestanda) låg på mycket acceptabla 98%.

År 2005 övertalade Donald Reinertsen mig att införa ett fullt utvecklat Kanban-system. Jag fick möjligheten 2006 när jag tog hand om programvaruavdelningen på Corbis i Seattle. 2007 började jag redovisa resultaten. Den första presentationen var på Lean New Product

Development Summit i Chicago i maj 2007. Jag följde upp det med ett öppet möte på Agile 2007 i Washington DC i augusti samma år. 25 personer deltog och tre av dem var från Yahoo!: Aaron Sanders, Karl Scotland och Joe Arnold. De åkte hem till Kalifornien, Indien och Storbritannien och införde Kanban i sina team som redan kämpade med Scrum. De startade också en diskussionsgrupp på Yahoo! som i skrivande stund har nästan 800 medlemmar. Kanban började spridas och ”early adpoters” pratade om sina erfarenheter.

Nu, 2009, ökar införandet av Kanban och allt fler rapporter strömmar i. Vi har lärt oss mycket om Kanban under de senaste 5 åren och vi fortsätter att lära oss mer varje dag. Jag har fokuserat mitt eget arbete på att utföra Kanban, skriva om Kanban, tala om Kanban och tänka på Kanban för att bättre förstå och förklara det för andra. Jag har medvetet tagit avsteg från att jämföra Kanban med befintliga agila metoder, även om visst arbete ägnades 2008 åt att förklara varför Kanban förtjänade att betraktas som en agil-kompatibel metod

Jag har överlämnat åt andra med större erfarenhet att svara på frågor som ”Hur är Kanban jämfört med Scrum?” Jag är så nöjd att Henrik Kniberg och Mattias Skarin har dykt upp som ledare i detta avseende. Du, kunskapsarbetare i ämnet, behöver information för att fatta välgrundade beslut och komma vidare i arbetet. Henrik och Mattias servar dina behov på ett sätt som jag aldrig skulle kunna. Jag är särskilt imponerad av Henriks genomtänkta jämförelsestrategi och hans sakliga och icke påstridiga, balanserade framställning. Hans teckningar och illustrationer är särskilt insiktsfulla och besparar dig ofta från att läsa åtskilliga sidor text. Mattias fallstudie är viktigt eftersom det visar att Kanban är mycket mer än teori och det visar genom exempel på vilket sätt det kan vara användbart för dig i din organisation.

Jag hoppas att du tycker om denna bok som jämför Kanban med Scrum och att det ger dig större insikt i agilt i allmänhet och både Kanban och Scrum i synnerhet. Om du vill veta mer om Kanban besök vår community, The Limited WIP Society, http://www.limitedwipsociety.org/

David J. Anderson

Sequim, Washington, USA
8 juli, 2009

Introduktion

I vanliga fall skriver vi inte böcker. Vi föredrar att lägga vår tid djupt nedgrävda i utmaningen att hjälpa kunder att optimera, debugga och refaktorisera sina utvecklingsprocesser och organisationer. På senare tid har vi dock noterat en trend och vi skulle vilja dela med oss av våra tankar kring det. Här är en typisk situation:

Helena “Nu har vi äntligen fått fullt grepp om Scrum!”

Peter ”Går det bra då?”

Helena ”Ja, det är ju bättre än det vi hade innan…”

Peter “Men…?”

Helena ”…jo, du förstår, vi har börjat med support och underhåll nu…”

Peter “Ja, och…?”

Helena ”Ja, alltså, vi gillar allt det där med att sorteringsprioritera behovslistan, självorganiserande utvecklingsteam, dagliga avstämningsmöten, återblick, och så vidare…”

Peter “Så vad är problemet?”

Helena ”Vi misslyckas med målen i varenda sprint.”

Peter “Varför då?”

Helena ”För att vi tycker det är svårt att binda oss till en 2-veckorsplan. Iterationerna har ingen speciell betydelse längre. Vi bara jobbar på med det som är viktigast för dagen. Vi kanske skulle köra 1-veckors-sprintar istället?”

Peter “Skulle ni kunna binda er till en veckas arbete? Har ni möjlighet att fokusera och arbeta i fred i en vecka?”

Helena ”Nej, inte direkt. Det dyker ju upp saker hela tiden. Kanske endags-sprintar…”

Peter “Tar era uppgifter mindre än en dag att utföra?”

Helena ”Nä, ibland tar de flera dagar…”

Peter “Så endags-sprintar skulle inte heller fungera. Har ni funderat på att skippa sprintar helt och hållet?”

Helena ”Ärligt talat, ja, det skulle vi vilja. Men är inte det emot principerna i Scrum?”

Peter “Scrum är bara ett verktyg. Ni bestämmer själva hur och när ni använder det. Låt er inte styras av verktyget.”

Helena ”Men vad ska vi göra då?”

Peter “Har du hört talas om Kanban?”

Helena ”Vad är det? Vad är det för skillnad mellan det och Scrum?”

Peter ”Här. Läs den här boken.”

Helena ”Men jag gillar ju resten av Scrum. Måste vi byta nu?”

Peter ”Nä, man kan kombinera metoderna!”

Helena ”Va? Hur då?”

Peter ”Läs bara vidare…”

Syfte med denna bok

Om du är intresserad av agil systemutveckling har du förmodligen hört talas om Scrum och du kan också ha hört talas om Kanban. En fråga som vi hör allt oftare är ”så vad är Kanban, och hur fungerar det i jämförelse med Scrum?” Var kompletterar de varandra? Finns det några potentiella konflikter?

Syftet med denna bok är att lätta dimman så att du kan räkna ut hur Kanban och Scrum kan vara användbart i din situation.

Berätta för oss om vi lyckas!

Del I – Jämförelse

Den första delen av den här boken är ett försök att göra en objektiv och praktisk jämförelse mellan Scrum och Kanban. Det är en något uppdaterad version av originalartikeln "Kanban vs. Scrum" från april 2009. Den artikeln blev populär så jag bestämde mig för att göra en bok av den och be min kollega Mattias att krydda den med en "from the trenches"-fallstudie från en av våra kunder. Bra grejer! Hoppa gärna direkt till del II om du vill börja med fallstudien. Jag tar inte illa upp.

Ok, lite då kanske.

/Henrik Kniberg

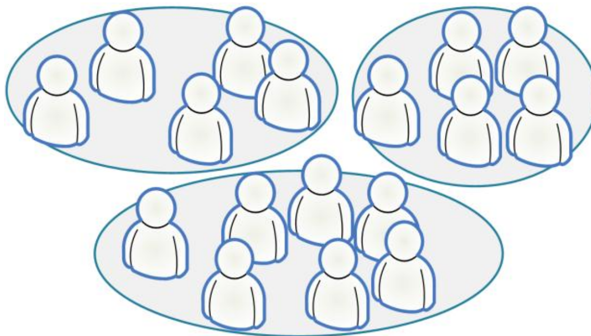
1

Vad är Scrum och Kanban egentligen?

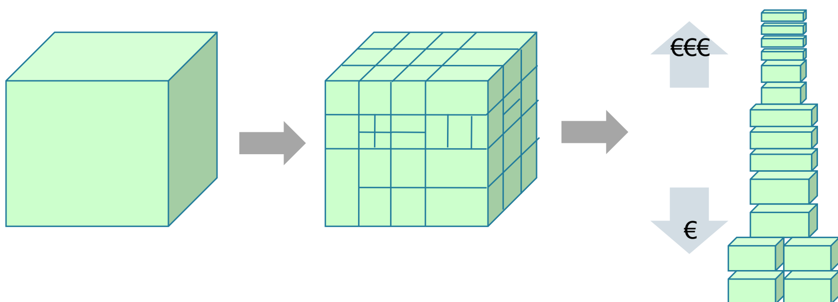
Ok, låt oss försöka summera Scrum och Kanban på mindre än 100 ord var.

Scrum i ett nötskal

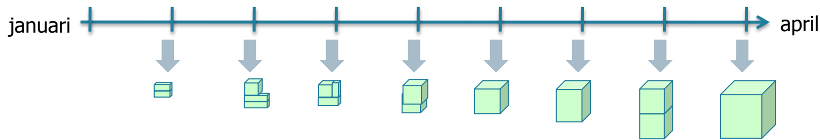
- **Dela upp organisationen** i små tvärfunktionella, självorganiserande lag.



- **Dela upp arbetet** i en lista med små, konkreta resultat. Sortera listan efter prioritet och uppskatta den relativa insats som krävs för varje post.



- **Dela upp tiden** i korta iterationer med fast längd (vanligen 1 – 4 veckor) med potentiellt levererbar kod som demonstreras efter varje iteration.



- **Optimera leveransplanen** och uppdatera prioriteringarna i samråd med kunden baserat på insikter som framkommit från granskning av leveransen efter varje iteration.

- **Optimera processen** genom återblick efter varje iteration.

Så istället för att en **stor grupp** spenderar **lång tid** på att bygga en **stor sak** har vi ett **litet lag** som tillbringar en **kort tid** med att bygga en **liten sak** men som **integrerar regelbundet** för att se helheten.

122 ord... nära nog.

Kolla in "Scrum och XP från Trenches" för mer information. Boken är gratis att läsa på nätet. Jag känner författaren och det är en trevlig kille :o).

http://www.crisp.se/ScrumAndXpFromTheTrenches.html

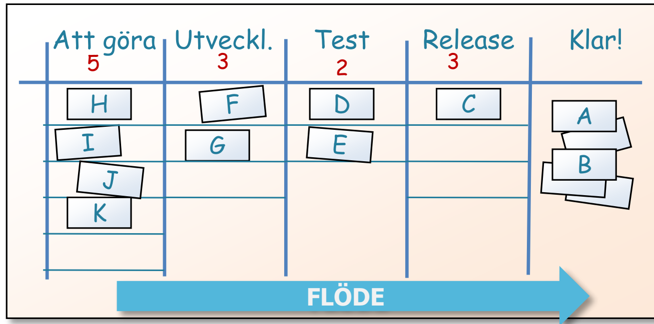
Kolla in http://www.crisp.se/scrum för fler länkar om Scrum.

Kanban i ett nötskal

- **Visualisera arbetsflödet**
    - Dela upp arbetet i bitar, skriv varje post på ett kort och sätt på väggen.
    - Använd namngivna kolumner för att illustrera var i flödet varje arbetspost befinner sig.

- **Begränsa pågående arbete (PA)** – tilldela tydliga gränser för hur många uppgifter som samtidigt kan pågå i varje arbetssteg.

- **Mät ledtiden** (genomsnittlig tid för att slutföra en arbetspost – kallas ibland “cykeltid”), optimera processen så att ledtiden blir så kort och förutsägbar som möjligt.



Användbara Kanban-länkar samlar vi på http://www.crisp.se/kanban.

2

Hur förhåller sig Scrum och Kanban till varandra?

Scrum and Kanban är processverktyg

Verktyg = vad som helst som används som ett medel för att utföra en uppgift eller för att uppnå ett syfte.

Process = hur du arbetar.

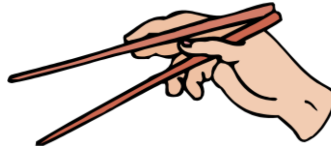
Scrum och Kanban är *processverktyg* i den meningen att de hjälper dig att arbeta mer effektivt genom att, till en viss grad, berätta för dig vad du ska göra. Java är också ett verktyg; det erbjuder ett enklare sätt att programmera en dator. En tandborste är ett annat verktyg; det hjälper dig att komma åt tänderna så att du kan rengöra dem.

Jämför verktyg för att förstå, inte för att döma

Kniv eller gaffel – vilket verktyg är bäst?



Rätt meningslös fråga, inte sant? Det beror ju på situation. Om du ska äta köttbullar så är förmodligen gaffeln bäst. För att hacka svamp så är nog kniven bäst. För att trumma på bordskanten så funkar nog vilken som. Om du ska äta en köttbit så vill du nog använda båda verktygen tillsammans. Till ris… tja… somliga föredrar gaffel, andra föredrar pinnar.



Så när vi jämför olika verktyg med varandra så bör vi vara försiktiga. Jämför för att förstå, inte för att döma.

Inget verktyg är fullständigt, inget verktyg är perfekt

Scrum och Kanban är, precis som alla andra verktyg, varken perfekta eller fullständiga. De talar inte om *allt* som du måste göra; de ger dig bara vissa begränsningar och riktlinjer. Scrum, till exempel, begränsar dig till att använda tidsbegränsade iterationer och tvärfunktionella lag. Kanban begränsar dig till att använda synliga tavlor och genom längden på kön.

Intressant nog så är värdet av ett verktyg just att det *begränsar dina möjligheter*. Ett processverktyg som tillåter dig att göra precis vad som helst är inte särskilt användbart. En sådan process skulle vi kunna kalla ”Gör hur du vill” eller varför inte ”Gör rätt”. ”Gör rätt”-processen fungerar garanterat; det är universallösningen! För om den inte fungerar så har du uppenbarligen inte följt processen :o)

Använder du rätt verktyg så kommer det hjälpa dig att lyckas men det kommer inte att garantera framgång. Det är lätt att förväxla ett lyckat/misslyckat *projekt* med ett lyckat/misslyckat *verktyg*.

- Ett projekt kan lyckas tack vare ett fantastiskt verktyg.
- Ett projekt kan lyckas trots ett värdelöst verktyg
- Ett projekt kan misslyckas på grund av ett värdelöst verktyg
- Ett projekt kan misslyckas trots ett fantastiskt verktyg

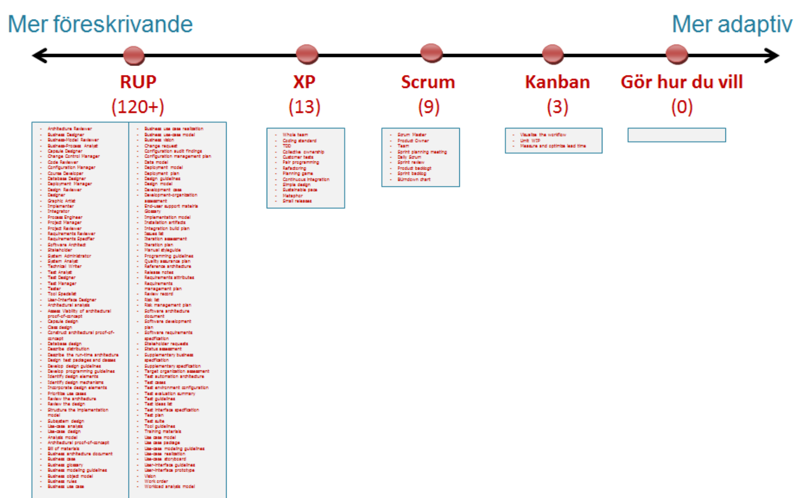
Scrum föreskriver mer än Kanban

Ett sätt att jämföra verktyg är hur många regler de har. *Föreskrivande* betyder ”fler regler att följa” och *adaptiv* betyder ”färre regler att följa”. 100 % föreskrivande betyder att du inte behöver använda hjärnan; det finns regler för allt. 100 % adaptiv betyder ”Gör hur du vill”, det finns inga regler eller begränsningar alls. Du håller säkert med om att extremerna är rätt löjliga.

Agila metoder kallas ibland för *lättviktiga* metoder just för att de är mindre föreskrivande än traditionella metoder. Faktum är att den första grundsatsen i manifestet för agil utveckling är ”individer och interaktioner framför processer och verktyg”.

Både Scrum och Kanban är väldigt adaptiva, men *relativt sett* är Scrum mer föreskrivande än Kanban. Scrum har fler begräsningar och ger därmed mindre utrymme för alternativ. Ett exempel är att Scrum föreskriver tidsbegränsade iterationer. Det gör inte Kanban.

Låt oss jämföra med några andra processverktyg på föreskrivande-adaptiv-skalan:



RUP är rätt så föreskrivande – den har över 30 roller, över 20 aktiviteter och över 70 dokument; en överväldigande massa saker att lära sig. Det är förvisso inte meningen att du ska lära dig allt. Tanken är att du ska välja ut en lämplig delmängd som passar projektet. Tyvärr verkar det vara rätt så svårt i praktiken: ”Hmmm… behöver vi dokumentet *Configuration audit findings*? Behöver vi rollen *Change control manager*? Ingen aning, det är bäst vi behåller dem uti fall att.” Det här kan vara en orsak till att RUP-anpassningar ofta blir rätt tungviktiga jämfört med agila metoder som Scrum och XP.

XP (eXtreme Programming) är rätt så föreskrivande jämfört med Scrum. Det omfattar det mesta i Scrum plus ytterligare ett antal rätt så specifika ingenjörsmetoder såsom testdriven utveckling och parprogrammering.

Scrum är mindre föreskrivande än XP eftersom det inte dikterar några speciella ingenjörsmetoder. Scrum är dock mer föreskrivande än Kanban eftersom det påbjuder iterationer och tvärfunktionella lag.

En av de största skillnaderna mellan Scrum och RUP är att i RUP får du för mycket och förväntas ta bort det du inte behöver. I Scrum får du för lite och förväntas lägga till det som saknas.

Kanban lämnar nästan allt öppet. Det enda tvingande är ”Visualisera arbetsflödet” och ”Begränsa PA”. Inte så långt från ”Gör hur du vill” men likväl förvånansvärt kraftfullt.

Begränsa dig inte till ett verktyg!

Blanda och kombinera verktyg som du vill! Jag kan till exempel knappast föreställa mig ett framgångsrikt Scrum-lag som inte också inkluderar de flesta delarna i XP. Många Kanban-lag använder dagliga ståuppmöten (en Scrum-sedvana). Vissa Scrum-lag skriver en del av uppgifterna i behovslistan (”backlog”) som användningsfall (en RUP-sedvana) eller begränsar kölängden (en Kanban-sedvana). Gör det som funkar för er.

Miyamoto Musashi, en samuraj som levde på 1600-talet och som var berömd för sin teknik att fäktas med två svärd, uttryckte det väl:



> Skapa inte tillgivenhet till ett enskilt stridsmedel eller en enskild stridsmetod.
>
> \- Miyamoto Musashi

Var dock uppmärksam på begränsningarna i varje verktyg. Om du exempelvis använder Scrum och bestämmer dig för att sluta med tidsbegränsade iterationer (eller någon annan central del av Scrum) så säg inte att du följer Scrum. Scrum är minimalistiskt som det är. Om du tar bort saker och fortfarande kallar det Scrum så blir begreppet så småningom

meningslöst och förvirrande. Kalla det ”Scrum-inspirerat” eller ”en delmängd av Scrum” eller varför inte ”Scrummigt” :o)

3

Scrum föreskriver roller

Scrum föreskriver 3 roller: produktägare (sätter produktvision och prioriteringar), lag (implementerar produken) och Scrum Master (tar bort hinder och ger processledning).

Kanban föreskrivr inte några roller alls.

Det betyder inte att du inte kan eller ska ha en produktägarroll i Kanban. Det betyder bara att du inte *måste*. I både Scrum och Kanban är det fritt fram att lägga till de roller du behöver.

Var dock försiktig när du lägger till roller. Säkerställ att ytterligare roller faktiskt ökar värdet och inte står i konflikt med andra delar i processen. Är du säker på att du behöver den där projektledarrollen? I ett stort projekt kanske det är en ypperlig idé, det är kanske den person som hjälper till att synka flera lag och produktägare med varandra. I ett litet projekt kan den rollen vara rent slöseri, eller ännu värre, kan leda till suboptimering och detaljstyrning.

Det generella tankesättet i både Scrum och Kanban är “less is more”. Så när du är osäker: börja med mindre.

I resten av boken använder jag termen “produktägare” för att representera den som sätter prioriteringarna för ett lag, oavsett vilket process som används.

4

Scrum föreskriver tidsbegränsade iterationer

Scrum baseras på tidsbegränsade iterationer. Du kan välja längd på iterationen men den allmänna idén är att bibehålla längden på varje iteration över en längre period och därmed etablera en *rytm*.

- **Början av en iteration:** En iterationsplan skapas, dvs. laget drar ett specifikt antal uppgifter från behovslistan för produkten baserat på produktägarens prioriteter och hur mycket laget tror att de kan slutföra i en iteration

- **Under iterationen:** Laget fokuserar på att slutföra de uppgifter som de har förbundit sig att slutföra. Iterationens omfattning är fixerat.

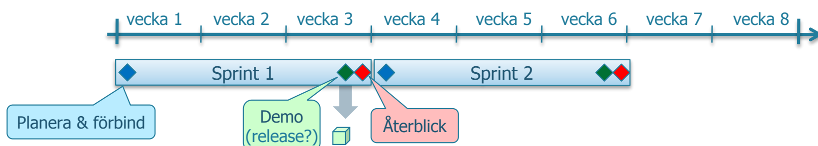
- **Slutet av en iteration:** Laget demonstrerar fungerande kod för relevanta intressenter. Idealt är detta *potentiellt leveransklar* kod (dvs. testad och färdig att köra). Därefter gör laget en återblick för att diskutera och förbättra processen.

En Scrum-iteration är alltså en enstaka tidsbegränsad *rytm* som kombinerar tre olika aktiviteter: planering, processförbättring och (idealt) leverans.

I Kanban föreskrivs inte tidsbegränsade iterationer. Du kan välja när du vill planera, processförbättra och leverera. Du kan välja att göra dessa aktiviteter regelbundet (”leverans varje måndag”) eller vid behov (”leverans när vi har något användbart att leverera”).

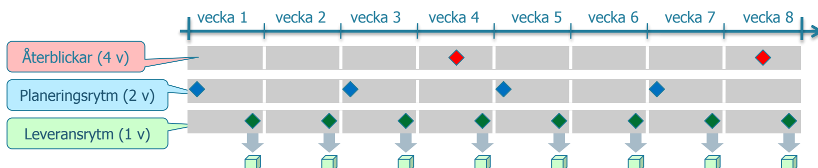
Lag #1 (enkel rytm)

“Vi gör Scrum-iterationer”



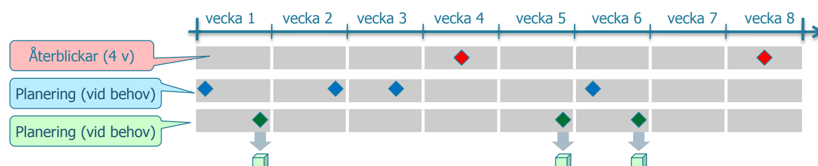
Lag #2 (tre rytmer)

"Vi har tre olika rytmer. Varje vecka släpper vi det som är redo för leverans. Varannan vecka har vi planeringsmöten och updaterar prioriteringar och leveransplaner. Var fjärde vecka har vi återblicksmöten för att justera och förbättra vår process"



Lag #3 (främst händelsestyrt)

"Vi kör igång ett planeringsmöte så fort vi börjar få ont om saker att göra. Vi påbörjar en leverans så fort det finns ett antal Minsta Säljbara Funktioner (MSF) som är klara för leverans. Vi startar en spontan kvalitetsdiskussion när vi stöter på samma problem två gånger. Vi gör också en mer djuplodande återblick var fjärde vecka."

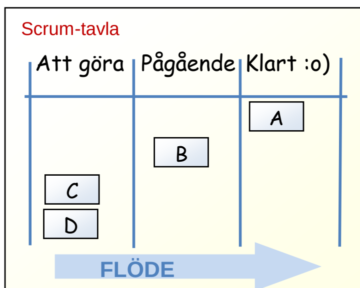


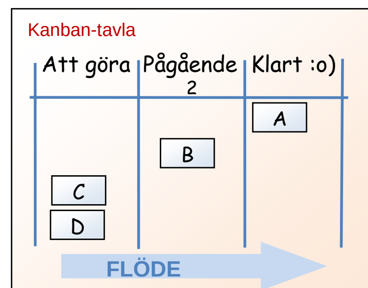
5

Kanban begränsar PA per tillstånd, Scrum begränsar PA per iteration

I Scrum visar sprintens uppgiftslista vilka uppgifter som ska slutföras i en pågående iteration (=”sprint” i Scrum). Detta representeras ofta med kort på en väggyta som kallas Scrum-tavla eller uppgiftstavla.

Så vad är skillnaden mellan en Scrum-tavla och en Kanban-tavla? Vi börjar med ett trivialt projekt och jämför de båda:





I båda fall håller vi koll på ett antal uppgifter som rör sig framåt i ett arbetsflöde. Vi har valt tre tillstånd: att göra, pågående och klart. Du kan välja vilka tillstånd du vill – vissa lag lägger till tillstånd som integration, test, leverans, etc. Glöm dock inte ”less is more”-principen.

Vad är då skillnaden mellan de här två exempeltavlorna? Jepp – den lilla 2:an i den mellersta kolumnen på Kanban-tavlan. Det är allt. Den 2:an betyder ”det får inte finnas mer än 2 uppgifter i den här kolumnen i varje givet ögonblick”.

I Scrum finns det ingen regel som hindrar laget att placera alla uppgifter i pågående-kolumnen samtidigt. Det finns dock en implicit gräns eftersom iterationen har en fixerad omfattning. I det här fallet är den implicita gränsen 4 per kolumn eftersom det bara finns 4 uppgifter på hela tavlan. Så Scrum begränsar PA indirekt medan Kanban gör det direkt.

Tids nog lär sig de flesta Scrum-lag att det är en dålig idé att ha för många samtidigt pågående uppgifter och utvecklar en kultur där man försöker

slutföra pågående uppgifter innan nya påbörjas. Det finns till och med de som beslutar att explicit begränsa antalet uppgifter som får finnas i pågående-kolumnen och så – tadaaa! – Scrum-tavlan har blivit en Kanban-tavla!

Så både Scrum och Kanban begränsar PA men på olika sätt. Scrum-lag mäter vanligtvis *hastighet* – hur många uppgifter (eller motsvarande enheter såsom ”historiepoäng”) som slutförs per iteration. När laget känner till sin hastighet så blir det deras PA-gräns (eller åtminstone en vägledning). Ett lag som har en snitthastighet på 10 kommer vanligtvis inte ta in mer än 10 uppgifter (eller story-poäng) i en sprint.

Så i Scrum *begränsas PA per tidsenhet.*

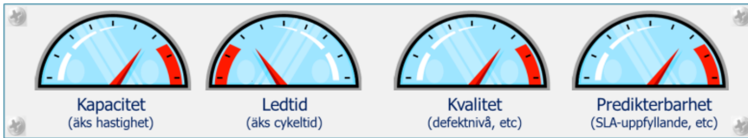
I Kanban *begränsas PA per tillstånd.*

I Kanban-exemplet ovan får det finnas högst 2 uppgifter i tillståndet ”pågående” vid varje givet ögonblick i tiden, oavsett vad rytmen är. Du måste välja vilken gräns som gäller för vilket tillstånd men den allmänna idén är att begräsna PA för *samtliga* tillstånd, med början så tidigt som möjligt och avslut så sent i värdekedjan som möjligt. I exemplet ovan borde vi överväga att även sätta en PA-gräns för ”att göra”-kolumnen (eller vad du nu kallar in-kön). När vi har satt PA-gränserna kan vi börja mäta och förutsäga ledtid, dvs. snittiden för en uppgift att förflytta sig hela vägen över tavlan. Med förutsägbara ledtider kan vi förbinda oss till SLA (service-level agreements) och göra realistiska leveransplaner.

Om storleken på uppgifterna varierar kraftigt kan du överväga att istället sätta PA-gränser i termer av story-poäng eller någon annan enhet som du använder. Det finns lag som investerar tid i att bryta ner uppgifter till ungefär samma storlek och minska tiden som läggs på att estimera saker (man kan till och med anse tidsestimering som slöseri). Det är lättare att skapa ett välflytande system om uppgifterna är av ungefär samma storlek.

6

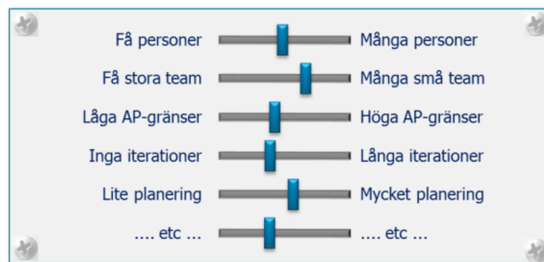
Båda är empiriska



Tänk om det fanns reglage för mätarna ovan och att du kunde konfigurera din process bara genom att vrida på reglagen. ”Jag vill ha hög kapacitet, kort ledtid, hög kvalitet och hög predikterbarhet. Så jag vrider rattarna till 10, 1, 10 och 10.”

Skulle inte det vara toppen? Tyvärr finns det inga sådana direkta kontrollmekanismer. Inte vad jag vet i alla fall. Säg gärna till om du hittar några.

Vad vi istället har är ett gäng *indirekta* kontroller.



Både Scrum och Kanban är empiriska i det avseende att du förväntas experimentera med processen och anpassa den till din omgivning. Faktum är att du *måste* experimentera. Varken Scrum eller Kanban har alla svar – de ger dig bara ett antal grundläggande begränsningar för att tvinga dig till processförbättring.

- Scrum säger att man ska ha tvärfunktionella lag. Så vem ska vara i vilket lag? Vet inte, experimentera.

- Scrum säger att laget väljer hur mycket arbete som tas in i sprinten. Så hur mycket ska de ta in? Vet inte, experimentera.

- Kanban säger att du ska begränsa PA. Så vad är gränsen? Vet inte, experimentera.

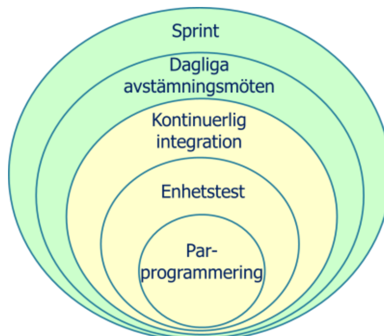
Som jag tidigare nämnt medför Kanban färre begränsningar än Scrum. Det betyder att du får fler parametrar att fundera kring, fler reglage att vidra på. Det kan vara både en nackdel och en fördel beroende på situation. När du öppnar dialogrutan med inställningar i en applikation, föredrar du då 3 justerbara alternativ eller 100 justerbara alternativ? Troligtivs något däremellan. Det beror på hur mycket du måste justera och hur väl du behärskar applikationen.

Låt oss anta att vi sänker PA-gränsen baserat på hypotesen att det kommer att förbättra processen. Vi mäter sedan hur saker som kapacitet, ledtid, kvalitet och predikterbarhet förändras. Vi drar slutsatser från resultaten och sedan ändrar vi på några andra saker och på sätt genomför vi ständiga förbättringar av processen.

Det finns många namn för det här. Kaizen (ständiga förbättringar på Lean-språk), Inspektera och Anpassa (Scrum-språk), Empirisk Processkontroll, eller varför inte Den Vetenskapliga Metoden.

Den mest kritiska delen i det här är *återkopplingsloopen*. Ändra något => Ta reda på hur det gick => Lär av det => Ändra något igen. Generellt sett vill du ha en så kort återkopplingsloop som möjligt så att du kan anpassa processen snabbt.

I Scrum är sprinten den grundläggande återkopplingsloopen. Det finns dock flera, särskilt om du kombinerar det med XP:



När det görs rätt så ger Scrum + XP en mängd värdefulla återkopplingsloopar.

Den innersta återkopplingsloopen, parprogrammering, har en återkopplingstid på ett par sekunder. Defekter hittas och rättas inom några sekunder från att de skapas (”Vänta, skulle inte den där variablen vara 3?”). Det här är ”bygger vi det **rätt**?”-återkopplingen.

Den yttre återkopplingsloopen, sprinten, ger oss en återkopplingstid på ett par veckor. Det här är ”bygger vi **rätt** saker?”-återkopplingen.

Kanban då? Tja, först och främst kan du (och borde nog) stoppa in alla ovanstående feedbackloopar i processen oavsett om du använder Kanban eller ej. Kanban ger dig sedan några mycket användbara realtidsmätetal:

- Genomsnittlig ledtid. Uppdateras varje gång en uppgift når “Klart” (eller vad du nu kallar den högersta kolumnen).
- Flaskhalsar. Typiska symptom är att kolumn X är fylld med uppgifter samtidigt som kolumn X+1 är tom. Leta efter ”luftbubblor” på tavlan.

Det trevliga med realtidsmätningar är att du kan välja längd på din återkopplingsloop baserat på hur ofta du önskar analysera mätningarna och göra förändringar. För lång återkopplingsloop innebär att din processförändring blir långsam. För kort återkopplingsloop innebär att processen inte hinner stabilisera sig mellan förändringarna vilket kan leda till sammanbrott.

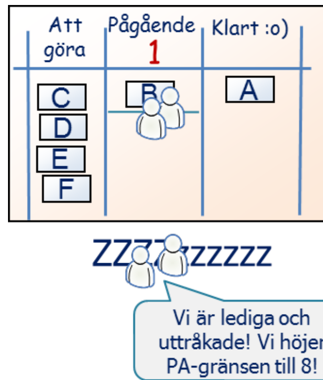
Faktum är att längden på återkopplingsloopen är en av de saker du kan experimentera med... som en slags meta-återkopplingsloop.

OK, jag slutar nu.

Exempel: Experimentera med PA-gränser i Kanban

En av de typiska “justeringspunkterna” i Kanban är PA-gränsen. Så hur vet man att man fått rätt nivå?

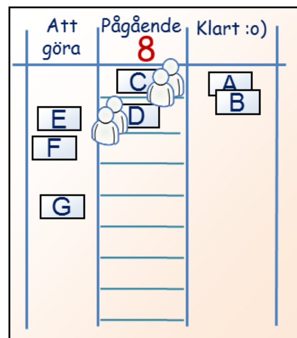
Anta att vi har ett lag med 4 personer och att vi bestämmer oss för att börja med en PA-gräns på 1.



Så fort vi börjar arbeta på en uppgift så får vi inte börja på en ny uppgift förrän den första är klar. Så den kommer att bli gjord rätt fort.

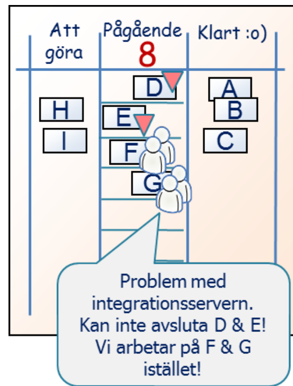
Toppen! Men sedan visar det sig att det inte är rimligt att alla 4 ska arbeta på samma uppgift (i den här enkla situationen), så vi har alltså personer som sitter sysslolösa. Om det bara händer då och då så är det inga problem men om det händer regelbundet blir konsekvensen att den genomsnittliga ledtiden kommer att öka.

Så om en PA-gräns på 1 var för långsamt, hur vore det att öka den till 8?

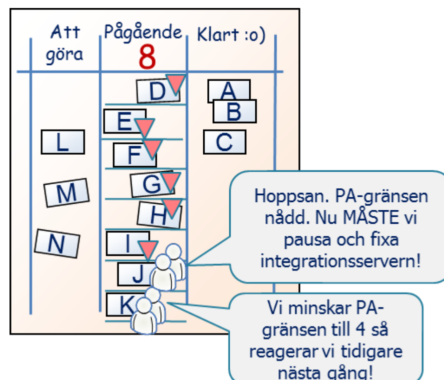


Det funkar bättre en stund. Vi upptäcker att, i genomsnitt, så får vi mer gjort om vi arbetar i par. Så med ett 4-personerslag har vi vanligtvis 2 pågående uppgifter vid varje givet ögonblick. En PA-gräns på 8 är bara en övre gräns så att ha färre pågående uppgifter går utmärkt!

Ponera nu att vi får problem med integrationsservern så att vi inte kan slutföra några uppgifter (vår definition av ”Klart” inbegriper integration). Sånt händer ibland, eller hur?

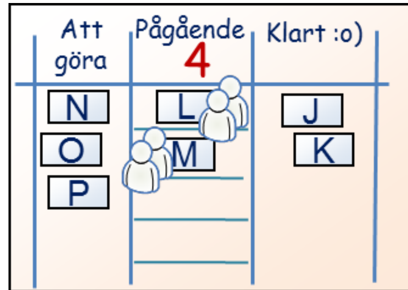


Eftersom vi inte kan slutföra uppgift D och E börjar vi arbeta på uppgift F. Vi kan inte integrera den heller så vi drar en ny uppgift G. Efter ett tag når vi Kanbangränsen – 8 uppgifter i ”Pågående”.



Vid det här laget kan vi inte ta in fler uppgifter. Hörrni, det är bäst att vi fixar den där förbaskade integrationsservern! PA-gränsen har fått oss att reagera och att lösa flaskhalsen istället för att bara lägga en massa oavslutat arbete på hög.

Det är bra. Men om PA-gränser var 4 så hade vi reagerat mycket tidigare och därmed fått en bättre genomsnittlig ledtid. Det är en balansgång. Vi mäter genomsnittlig ledtid och fortsätter att optimera vår PA-gräns för att optimera ledtiden:



Efter ett tag upptäcker vi kanske att uppgifter läggs på hög i ”Att göra”. Kanske är det dags att sätta en PA-gräns där också då.

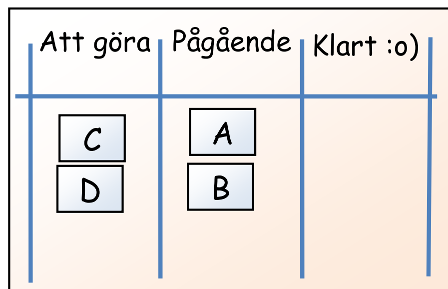
Varför behöver vi förresten en “Att göra”-kolumn? Tja, om kunden alltid fanns tillgänglig och kunde berätta för laget vad de ska göra härnäst så fort de undrar då skulle ”Att göra”-kolumnen inte behövas. Men i det här fallet är kunden inte alltid tillgänglig så ”Att göra”-kolumnen ger teamet en liten buffert att plocka arbete från.

Experimentera! Eller, som Scrumologerna säger, Inspektera och Anpassa!

7

Scrum motstår förändringar inom en iteration

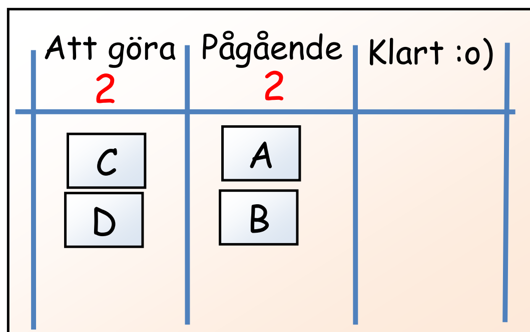
Anta att din Scrum-tavla se ut så här:



Vad händer om någon dyker upp och vill lägga till E till tavlan?

Ett Scrum-lag kommer vanligtvis att säga något i stil med ”Nej, tyvärr, vi har förbundit oss att göra A+B+C+D i den här sprinten. Men var så god att lägg till E till behovslistan för produkten. Om produktägaren tycker att den har hög prioritet så kommer vi att ta in den i nästa sprint.” Sprintar med rätt längd ger laget precis tillräckligt med fokuserad tid för att få något gjort samtidigt som produktägaren regelbundet tillåts hantera och uppdatera prioriterigar.

Så vad gör då ett Kanban-lag?



Ett Kanban-lag kanske säger “Var så god och lägg till E till Att göra-kolumnen. Maxgränsen är dock 2 uppgifter för den kolumnen så du måste i så fall ta bort C eller D. Vi arbetar just nu på A och B men så fort vi har kapacitet kommer vi att plocka in den översta uppgiften från Att göra”.

Så svarstiden (tiden det tar att reagera på förändringar i prioritering) i ett Kanban-lag är den tid det tar tills kapacitet blir tillgänglig, enligt den allmänna principen ”en uppgift ut = en uppgift in” (drivs av PA-gränserna).

I Scrum är responstiden i genomsnitt halva sprint-längden.

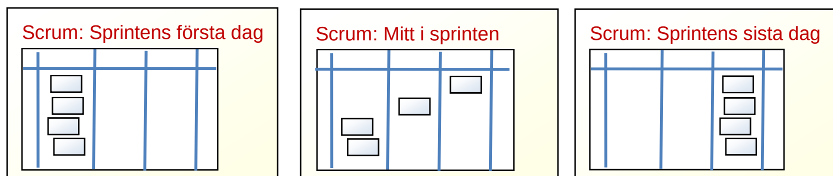
I Scrum kan inte produktägaren röra Scrum-tavlan eftersom laget har förbundit sig till ett specifikt antal uppgifter i iterationen. I Kanban måste du etablera dina egna spelregler för vem som får ändra vad på tavlan. Typiskt ger man produktägaren någon slags ”Att göra”-, ”Redo”-, ”Uppgifter”- eller ”Föreslagna”-kolumn längst till vänster där hon kan göra ändringar när hon vill.

De här två ansatserna är dock inte uteslutande. Ett Scrum-lag *får* besluta att produktägaren tillåts ändra prioriteringar under pågående sprint (trots att det normalt anses vara ett undantag). Och ett Kanban-lag *får* besluta att införa restriktioner för när prioriteringar får ändras. Ett Kanban-lag kan till och med bestämma sig för att använda tidsbegränsade iterationer med fastställt åtagande precis som i Scrum.

8

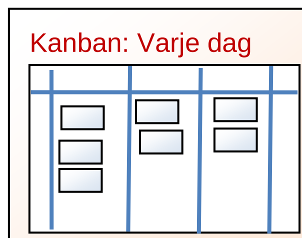
Scrum-tavlan nollställs efter varje iteration

I olika stadier av en sprint ser en Scrum-tavla typiskt ut så här.



När en sprint är avklarad rensas tavlan – alla uppgifter tas bort. En ny sprint påbörjas och efter sprintplaneringsmötet har vi en ny Scrum-tavla med nya uppgifter i kolumnen längst till vänster. Tekniskt sett är detta slöseri men för erfarna Scrum-lag tar detta normalt inte särskilt lång tid och rutinen att nollställa tavlan kan ge en trevlig känsla av fullbordan och avslut. Lite som att diska efter middagen – att göra det är en pina men det känns bra efteråt.

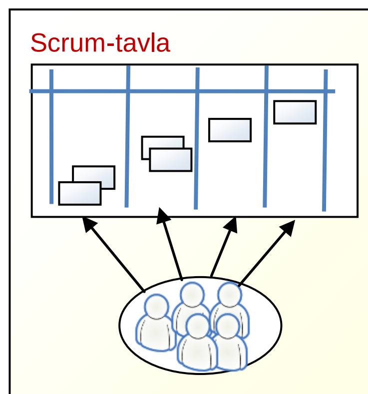
I Kanban är tavlan normalt en beständig sak – du behöver inte nollställa den och börja om.



9

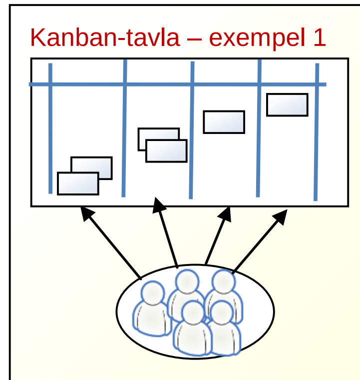
Scrum föreskriver tvärfunktionella lag

En Scrum-tavla ägs av exakt ett lag. Ett Scrum-lag är tvärfunktionellt, det består av alla de kompetenser som behövs för att färdigställa alla uppgifter i sprinten. En Scrum-tavla är vanligtvis synlig för den som är intresserad men bara det ägande Scrum-laget får ändra i den – det är deras verktyg för att hantera sitt åtagande för pågående iteration.

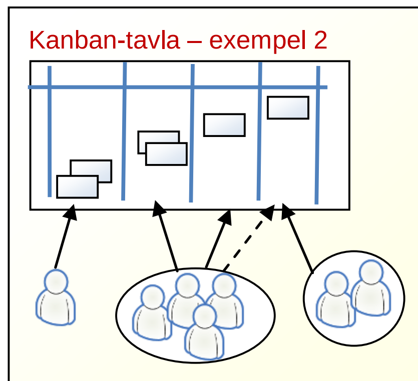


I Kanban är tvärfunktionella lag valfritt och tavlan behöver inte ägas av ett specifikt lag. En tavla relaterar till ett arbetsflöde, inte nödvändigtvis ett lag.

Här är två exempel:



Exempel 1: Hela tavlan betjänas av ett tvärfunktionellt lag. Precis som i Scrum.



Exempel 2: Produktägaren sätter prioriteringar i kolumn 1. Ett tvärfunktionellt lag utför utveckling (kolumn 2) och test (kolumn 3). Leverans (kolumn 4) görs av specialistlag. Det finns ett litet överlapp av kompetenser så om leveranslaget blir en flaskhals kan en av utvecklarna hjälpa dem.

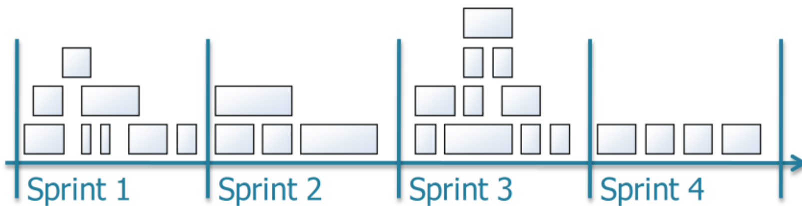
Så i Kanban måste du etablera ett antal spelregler för vem som får använda tavlan och hur. Sedan experimenterar du med reglerna för att optimera flödet.

10

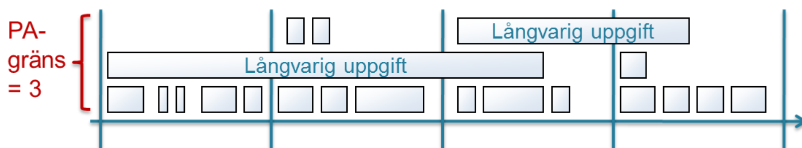
Scrum-uppgifterna måste få plats i en sprint

Både Scrum och Kanban är baserade på inkrementell utveckling, dvs. att bryta ner arbetet i mindre delar.

Ett Scrum-lag förbinder sig bara till de uppgifter som de tror att de kan klara av inom en iteration (baserat på definitionen av ”Klart”). Om en uppgift är för stor för att få plats i en sprint försöker laget och produktägaren hitta sätt att bryta ner den i mindre delar tills det får plats. Om uppgifterna tenderar att vara stora så blir iterationerna längre (men vanligtvis inte längre än 4 veckor).



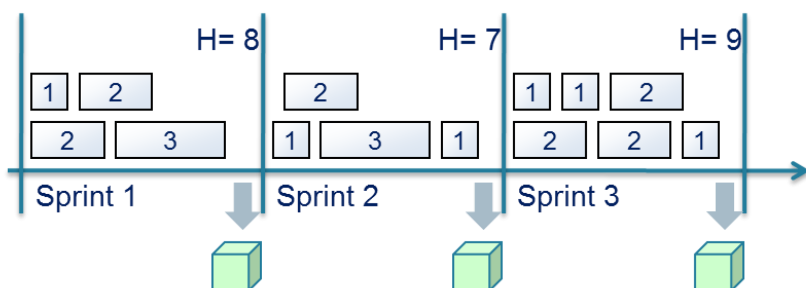
Kanban-lag försöker minimera ledtid och reglera flödet, vilket indirekt ger incitament att bryta ner uppgifter i relativt små bitar. Men det finns ingen explicit regel som säger att uppgifter måste vara tillräckligt små för att få plats i en specifik tidslucka. På samma tavla kan vi ha en uppgift som tar 1 månad att slutföra och en annan uppgift som tar 1 dag.



11

Scrum föreskriver estimering och hastighet

I Scrum förväntas lagen uppskatta den relativa storleken (=mängd arbete) av varje uppgift som de förbundit sig att utföra. Genom att summera storlekarna av alla slutförda uppgifter vid slutet av sprinten får vi fram hastigheten. Hastigheten är ett mått på kapacitet – hur många grejer vi kan leverera per sprint. Här är ett exempel på ett lag med en snitthastighet på 8.



Att veta att snitthastigheten är 8 är trevligt eftersom vi då kan göra realistiska förutsägelser om vilka uppgifter vi kan slutföra i kommande sprintar och därmed kan vi göra realistiska leveransplaner.

Kanban föreskriver inte estimering. Så om du behöver göra åtaganden måste du bestämma hur du skapar förutsägbarhet.

Vissa lag väljer att göra estimat och mäta hastigheten precis som i Scrum. Andra lag väljer att hoppa över estimeringen men försöker bryta ner alla uppgifter till ungefär lika stora delar. De kan sedan enkelt mäta hastighet i termer av hur många uppgifter som slutförs per tidsenhet (exempelvis funktioner per vecka). Somliga lag grupperar uppgifter i Minsta Säljbara Funktioner (MSF, engelska MMF) och mäter genomsnittlig ledtid per MSF och använder det för att etablera SLAer (Service Level Agreements) – till exempel ”när vi åtar oss en MSF levereras den alltid inom 15 dagar”

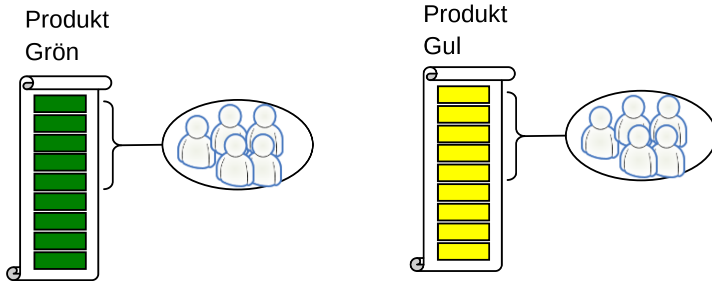
Det finna alla möjliga intressanta tekniker för Kanban-mässig leveransplanering och åtagandehantering – men det föreskrivs inga

specifika tekniker så Googla på bara och prova några olika tekniker tills du hittar en som passar din situation. Vi kommer nog att se några ”best practices” uppenbara sig med tiden.

12

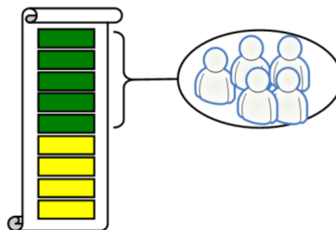
Båda tillåter arbete på flera produkter samtidigt

I Scrum är “behovslista för produkt” ett ganska olyckligt namn eftersom det antyder att alla behov måste gälla samma produkt. Här är två produkter, grön och gul, var och en med sin egen behovslista och sitt eget team:

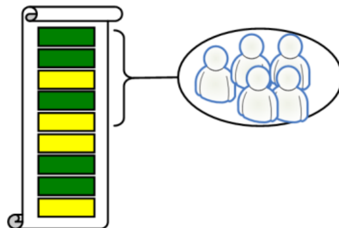


Om du bara har ett team då? Ja, tänk mer på behovslistan för produkten som en uppgiftslista för teamet. Den listar alla prioriteter för kommande iterationer för ett specifikt lag (eller uppsättning lag). Så om laget underhåller flera produkter kombinerar du listorna till en. Det tvingar oss att prioritera mellan produkter vilket i vissa fall är värdefullt.

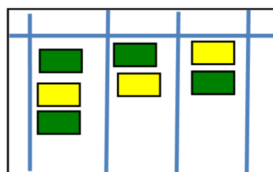
I praktiken kan man göra det här på flera sätt. En strategi kan vara att låta laget fokusera på en produkt per sprint:



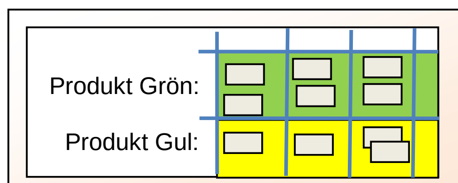
En annan strategi skulle kunna vara att låta laget arbeta på funktioner från båda produkter i varje sprint.



Samma sak gäller Kanban. Vi kan låta flera produkter flöda över samma tavla. Vi kan särskilja dem med olikfärgade kort:



…eller med "simbanor":



13

Båda är slimmade och lättrörliga

Jag tänker inte gå igenom ”Lean Thinking” och det agila manifestet men generellt sett är både Scrum och Kanban väl i linje med dess värderingar och principer. Exempelvis:

- Scrum och Kanban är båda så kallade dragsystem för planering vilket motsvarar lagerhanteringsprincipen JIT (Just In Time) i Lean. Det betyder att laget väljer när och hur många uppgifter de förbinder sig att göra – de ”drar” arbete när de är färdiga snarare än att det ”trycks” in utifrån. Precis som en skrivare som drar in nästa sida först när den är redo att skriva ut den (även om det är ett litet och begränsat parti papper som den kan dra ifrån).

- Scrum och Kanban är baserade på kontinuerlig och empirisk processoptimering vilket motsvarar Kaizen-principen i Lean.

- Scrum och Kanban betonar respons på förändringar framför att följa en plan (även om Kanban typiskt medger snabbare respons än Scrum), vilket är ett av de fyra värdena i det agila manifestet.

... och så vidare.

Från ett perspektiv kan Scrum ses som ganska oslimmat eftersom det föreskriver satsvis bearbetning i tidsbegränsade iterationer. Men det beror på iterationslängden och vad man jämför med. Jämfört med en mer traditionell process där man kanske integrerar och levererar 2-4 gånger per år så är ett Scrum-lag som producerar leveransfärdig kod varannan vecka extremt slimmat.

Men om du sedan fortsätter att göra iterationerna kortare och kortare närmar du dig egentligen Kanban. Och när du börjar fundera på att göra iterationerna kortare än 1 vecka skulle du kunna överväga att skippa tidsbegränsade iterationer helt och hållet.

Jag har sagt det förut och jag säger det igen: experimentera tills du hittar ett sätt som fungerar! Och fortsätt sedan att exeperimentera :o)

14

Mindre skillnader

Här är några skillnader som förefaller mindre betydelsefulla jämfört med dem som redan nämnts. Det kan dock vara bra att känna till dem.

Scrum föreskriver en prioriterad behovslista för produkten

I Scrum utförs alltid prioritering genom att sortera behovslistan för produkten och ändringarna träder i kraft till nästa sprint (inte i den pågående). I Kanban kan du välja godtyckligt prioriteringssystem (eller inget alls) och ändringarna träder i kraft så snart det finns tillgänglig kapacitet (snarare än vid bestämda tider). Det kanske finns eller inte finns en behovslista för produkten och den kan vara eller är inte prioriterad.

I praktiken spelar det här mindre roll. På en Kanban-tavla fyller den vänstra kolumnen vanligtvis samma syfte som behovslistan i Scrum. Oavsett om listan är sorterad efter prioritet eller inte så måste laget ha någon slags beslutsregel för vilken uppgift som ska tas först. Några exempel på sådana beslutsregler:

- Ta alltid den översta uppgiften.
- Ta alltid den äldsta uppgiften (varje uppgift har alltså en tidsstämpel).
- Ta vilken uppgift som helst.
- Spendera runt 20% av tiden på uppgifter som rör underhåll och 80% på uppgifter som rör ny funktionalitet.
- Dela upp lagets kapacitet ungefär jämnt mellan produkt A och produkt B.
- Ta alltid röda uppgifter först – om det finns några.

I Scrum kan behovslistan för produkten också användas mer Kanban-aktigt. Vi kan begränsa dess storlek och skapa beslutsregler för hur den ska prioriteras.

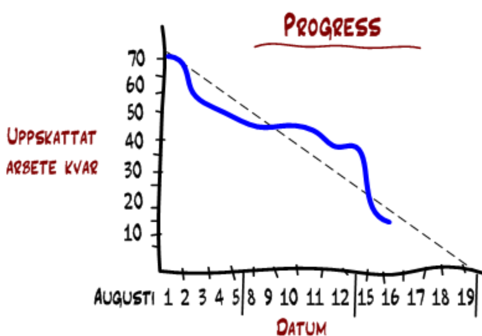
I Scrum föreskrivs dagliga möten

Ett Scrum-lag har korta möten (högst 15 minuter) varje dag vid samma tid och samma plats. Syftet med mötena är att sprida information om vad som pågår, att planera dagens arbete och att identifiera betydande problem. Det kallas ibland för dagliga stå-upp-möten eftersom de vanligtvis hålls stående (för att hålla det kort och för att bihålla en hög energinivå).

Dagliga stå-upp-möten föreskrivs inte i Kanban men de flesta Kanban-lag verkar göra det i alla fall. Det är en suverän teknik oavsett vilken process du använder.

I Scrum är mötets format personorienterat – varje person rapporterar, en efter en. Många Kanban-lag använder ett mer tavel-orienterat format med fokus på flaskhalsar och andra synliga problem. Den ansatsen är mer skalbar. Om du har 4 lag som delar samma tavla och som kör sina dagliga stå-upp-möten tillsammans så behöver vi inte nödvändigtvis stå och lyssna på var och en särskilt länge eftersom vi fokuserar på flaskhalsarna som syns på tavlan.

I Scrum föreskrivs progressdiagram



Progressdiagrammet för sprinten visar, på daglig basis, hur mycket arbete som återstår i den pågående iterationen.

Y-axelns enhet är densamma som för uppgifterna. Typiskt är det timmar eller dagar (om laget bryter ner uppgifterna till aktiviteter) eller ”historiepoäng” (om de inte gör det). Det finns dock många olika varianter av det här.

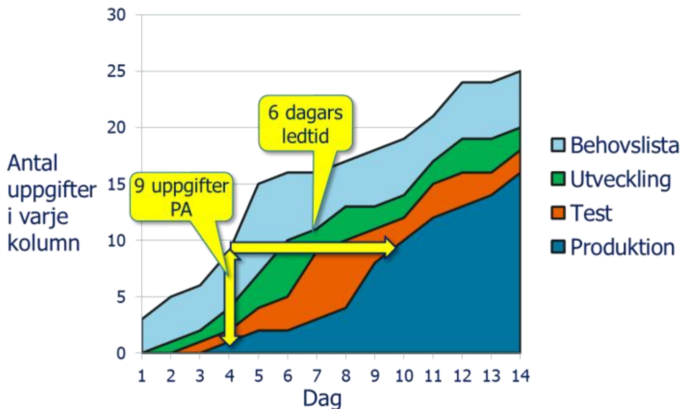
I Scrum används sprintens progressdiagram som ett av de viktigaste verktygen för att bevaka framsteg i en iteration.

Några lag använder också progressdiagram för leveranser vilket följer samma format men på leveransnivå – det visar vanligtvis hur många historiepoäng som återstår i behovslistan för produkten efter varje sprint.

Huvudsyftet med ett progressdiagram är att på ett enkelt sätt och så tidigt som möjligt ta reda på om vi ligger efter eller före i planen så att vi kan anpassa oss därefter.

I Kanban föreskrivs inte några progressdiagram. Faktum är att det inte föreskrivs något särskilt diagram alls. Men du får naturligtvis använda vilket diagram du vill (inklusive progressdiagram).

Här är ett exempel på ett kumulativt flödesdiagram. Den här typen av diagram illustrerar på ett trevligt sätt hur jämnt ditt flöde är och hur PA-gränserna påverkar ledtiden.



Så här funkar det. Varje dag summeras antalet uppgifter i respektive kolumn på Kanban-tavlan och värdena staplas ovanpå varandra utmed Y-axeln. Så på dag 4 fanns det 9 uppgifter på tavlan. Med början på den högersta kolumnen så fanns det 1 uppgift i Produktion, 1 uppgift i Test, 2 uppgifter i Utveckling och 5 uppgifter i Behovslistan. Om vi plottar dessa punkter varje dag och binder samman punkterna så får vi ett lika fint diagram som ovan. De vertikala och hortisontella pilarna illustrerar relationen mellan PA och ledtid.

Den horisontella pilen visar att uppgifter som lades till behovslistan på dag 4 i snitt tog 6 dagar att nå produktion. Omkring hälften av tiden var i Test. Vi kan se att om vi begränsade PA i Test och Behovslista skulle vi avsevärt minska den totala ledtiden.

Lutningen på den mörkblå ytan visar hastigheten (dvs. antalet uppgifter som driftsätts per dag). Över tiden kan vi se hur högre hastighet minskar ledtiden medan högre PA ökar ledtiden.

De flesta organisationer vill få saker gjort snabbare (=minska ledtiden). Olyckligtvis trillar många i fällan att anta att det betyder fler personer eller att fler ska arbeta övertid. Det mest effektiva sättet att få saker gjort snabbare är oftast att jämna ut flödet och begränsa arbetet till tillgänglig kapacitet, *inte* att lägga till fler människor eller att arbeta hårdare. Den här typen av diagram visar varför och ökar därmed sannolikheten för att laget och ledningen samarbetar mer effektivt.

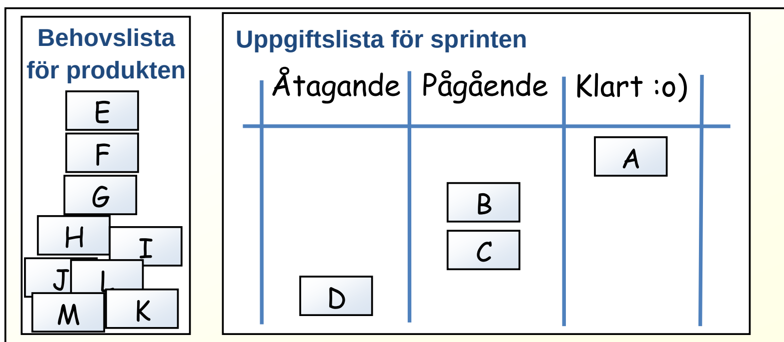
Det här blir ännu tydligare om vi skiljer på kötillstånd (som att ”vänta på test”) och arbetstillstånd (som att ”testa”). Vi vill definitivt minimera antalet uppgifter som står och väntar på kö och det kumulativa flödesdiagrammet hjälper oss att hitta rätt incitament för det.

15

Scrum-tavla vs. Kanban-tavla – ett mindre trivialt exempel

I Scrum är uppgiftslistan för sprinten bara en del av helheten – den del som visar vad laget gör i den pågående sprinten. Den andra delen är behovslistan för produkten – listan över saker som produktägaren vill få gjort i kommande sprintar.

Produktägaren får se men inte röra uppgiftslistan för sprinten. Hon kan göra förändringar i behovslistan för produkten när hon vill men ändringarna träder inte i kraft (dvs. ändringar av vilket arbete som blir gjort) förrän till nästa sprint.



När sprinten är färdig kan laget "leverera potentiellt levererbar kod" till produktägaren. Laget avslutar sprinten, genomför en sprintgranskning och demonstrerar stolt funktionerna A, B, C och D för produktägaren. Produktägaren kan nu besluta om detta ska levereras eller inte. Den sista delen – att faktiskt leverera produkten – ingår vanligtvis inte i sprinten och är därför inte synligt i uppgiftslistan för sprinten.

I samma scenario kan en Kanban-tavla istället se ut så här:

Behov	Utvalt 2	Utveckling 3		Driftas	Live!
		Pågående	Klart		
F G H I J L M K	D E	B C	A		X R Q

Nu är hela arbetsflödet på samma tavla – vi tittar inte bara på vad ett Scrum-lag gör i en iteration.

I exemplet ovan är “Behov”-kolumnen bara en generell önskelista utan särskild ordning. ”Utvalt”-kolumnen innehåller de högprioriterade behoven och har en Kanban-gräns på 2. Det får alltså bara finnas 2 högprioriterade uppgifter vid varje givet ögonblick. Så fort laget är redo att börja på en ny uppgift tar de den översta uppgiften från ”Utvalt”. Produktägaren kan göra ändringar i ”Behov”- och ”Utvalt”-kolumnerna när han vill men inte i de andra kolumnerna.

”Utveckling”-kolumnen (indelad i två underkolumner) visar vad som för närvarande utvecklas och den har en Kanban-gräns på 3. I nätverkstermer motsvarar Kanban-gränsen ”bandbredd” och ledtiden motsvarar ”pingtid” (eller svarstid).

Varför har vi delat in ”Utveckling”-kolumnen i de två underkolumnerna ”Pågående” och ”Klart”? Jo, för att ge driftsättningslaget en möjlighet att veta vilka uppgifter som de kan börja driftsätta.

Gränsen 3 för “Utveckling” delas mellan de två underkolumnerna. Varför då? Anta att det finns 2 uppgifter i ”Klart”:

Utveckling 3	
Pågående	Klart
B	A C

Det betyder att det bara får finnas 1 uppgift i "Pågående". Det betyder att det kommer att finnas överskottskapacitet – utvecklare som *skulle kunna* börja på en ny uppgift men som inte får det på grund av Kanban-gränsen. De ger dem ett starkt incitament att fokusera sin kraft på att hjälpa till att få saker i drift, rensa "Klart"-kolumnen och maximera flödet. Den här effekten är trevlig och gradvis – ju mer saker den finns i "Klart" desto mindre saker tillåts i "Pågående" vilket hjälper laget att fokusera på rätt saker.

Enuppgiftsflöde

Enuppgiftsflödet är ett slags "perfekt flöde"-scenario där varje uppgift flyter över tavlan utan att fastna i någon kö. Det betyder att i varje ögonblick finns det någon som arbetar på uppgiften. Så här kan tavlan se ut i en sådan situation:

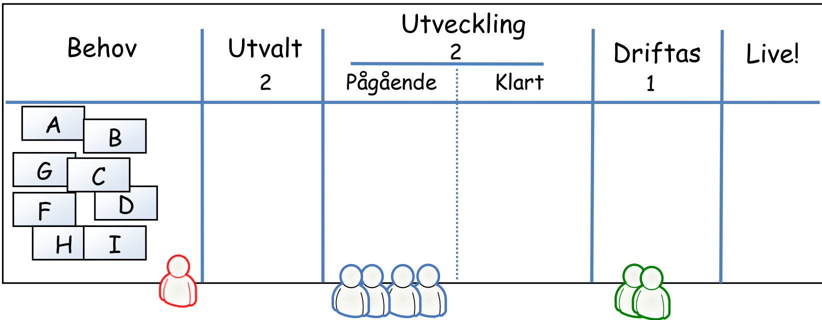
Behov	Utvalt 2	Utveckling 3: Pågående	Utveckling 3: Klart	I produktion :o)
		B	A	X, R, Q

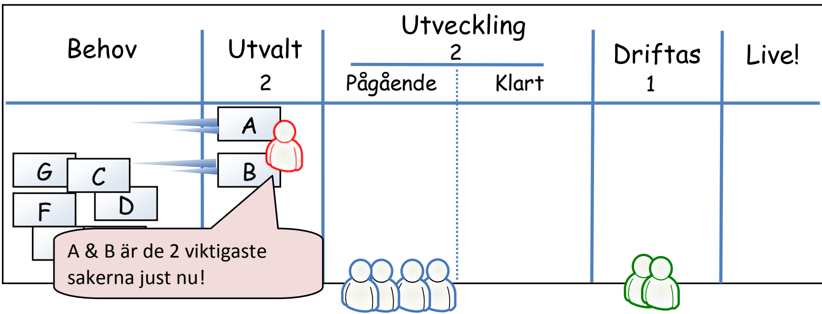
Just nu så utvecklas B samtidigt som A driftssätts. Så fort laget är redo för nästa uppgift frågar de produktägaren vad som är viktigast och får omedelbart svar. Om det här ideala scenariot fortsätter kan vi skrota de två köerna "Behov" och "Utvalt" och uppnå *riktigt* kort ledtid.

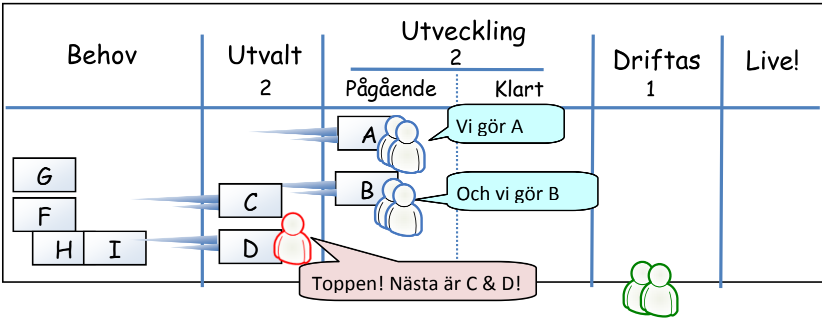
Corey Ladas (författare till "Scrumban", övers. anm.) har formulerat det väl: "Den ideala planeringsprocessen ska alltid förse utvecklingslaget med det bästa att arbeta på härnäst, varken mer eller mindre"´

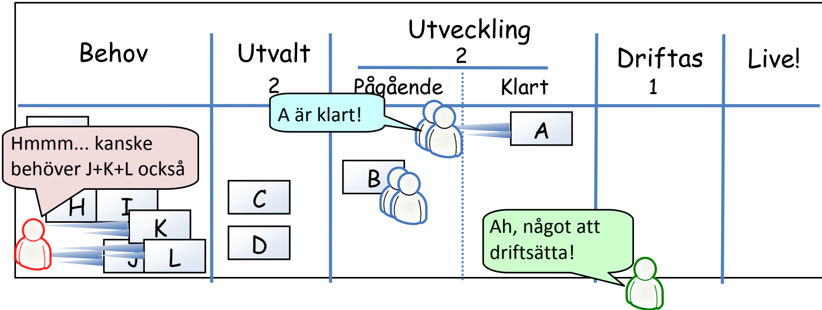
PA-gränserna används för att förhindra att problem går överstyr, så om allt flyter på används faktiskt inte PA-gränserna.

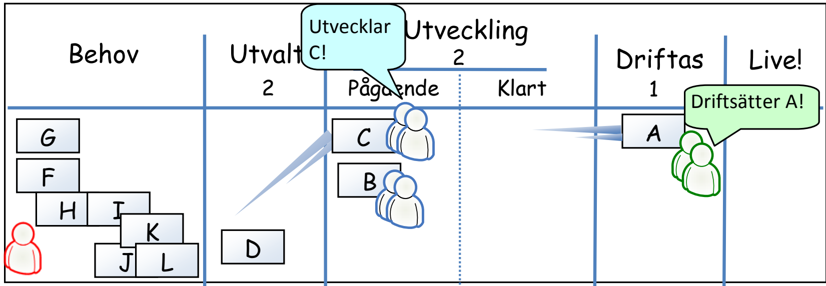
En dag i Kanban-land

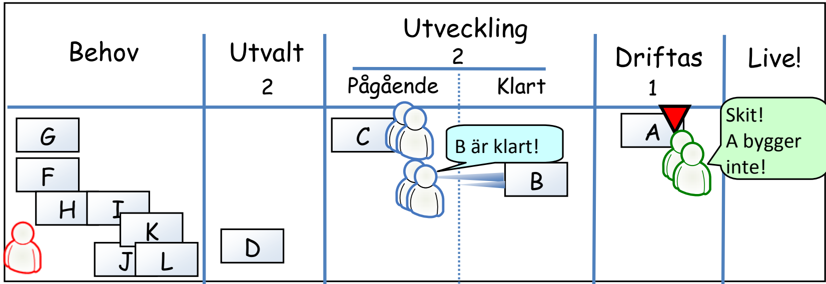


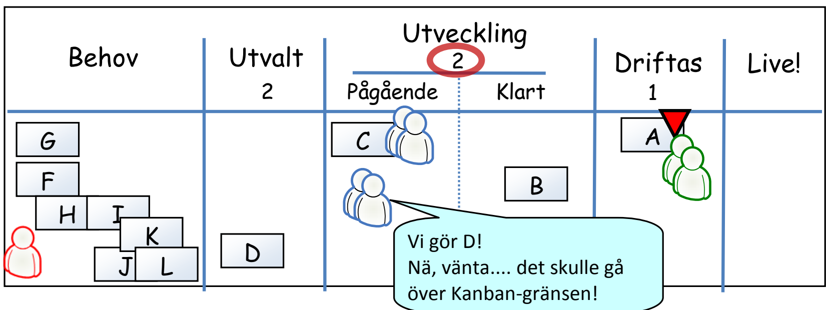


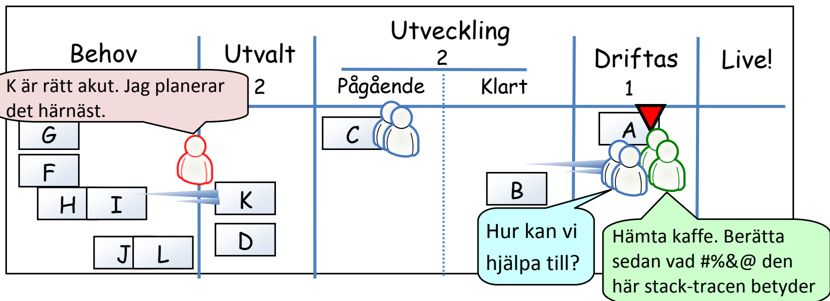


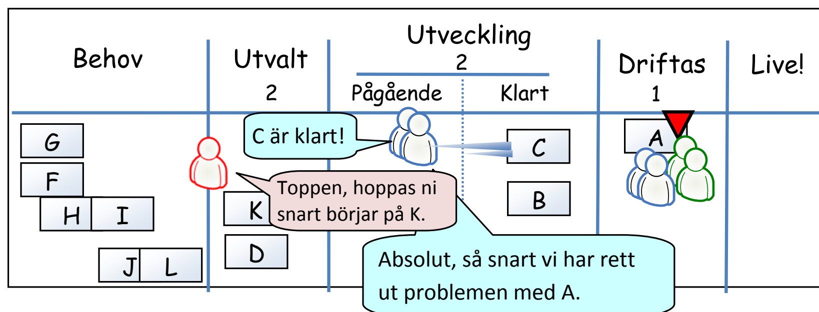


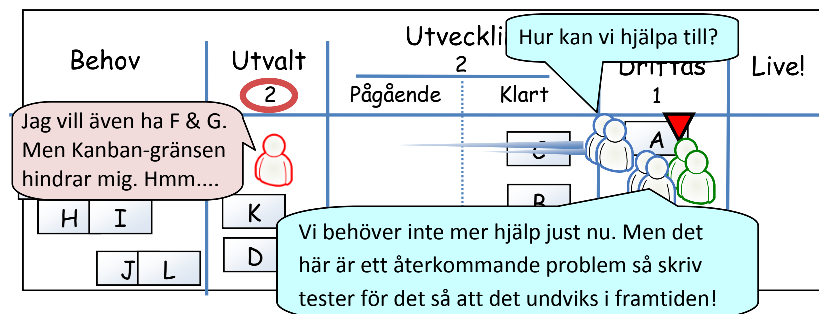

Behov
Utvalt
2
Utvecklar C!
Utveckling
2
Pågående
Klart
Driftas
1
Live!
Driftsätter A!
G
F
H
I
K
J
L
D
C
B
A

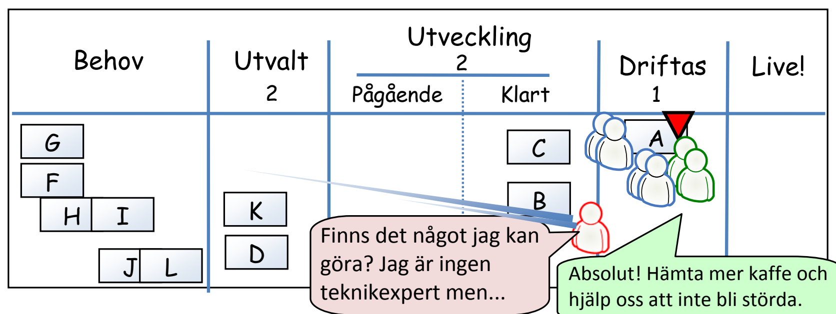

Behov
Utvalt
2
Utveckling
2
Pågående
Klart
Driftas
1
Live!
Skit!
A bygger inte!
B är klart!
G
F
H
I
K
J
L
D
C
B
A

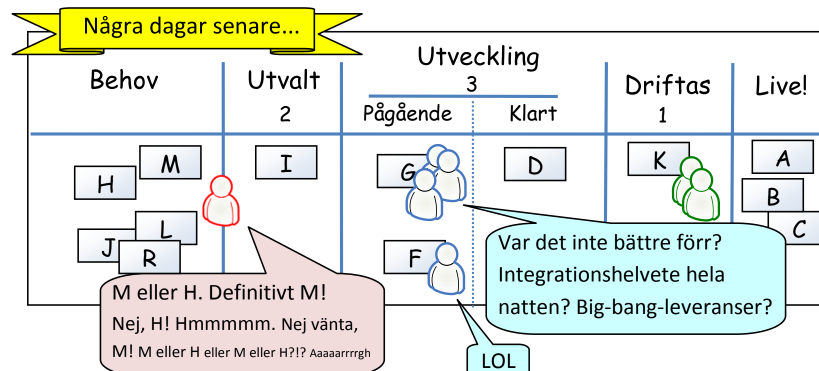

Behov
Utvalt
2
Utveckling
2
Pågående
Klart
Driftas
1
Live!
G
F
H
I
K
J
L
D
C
B
A
Vi gör D!
Nä, vänta.... det skulle gå över Kanban-gränsen!


Behov
Utvalt
2
Utveckling
2
Pågående
Klart
Driftas
1
Live!
K är rätt akut. Jag planerar det härnäst.
G
F
H
I
K
D
J
L
C
B
A
Hur kan vi hjälpa till?
Hämta kaffe. Berätta sedan vad #%&@ den här stack-tracen betyder


Behov
Utvalt
2
Utveckling
2
Pågående
Klart
Driftas
1
Live!
G
F
H
I
J
L
K
D
C
B
A
C är klart!
Toppen, hoppas ni snart börjar på K.
Absolut, så snart vi har rett ut problemen med A.


Behov
Utvalt
2
Utveckli
2
Pågående
Klart
1
Live!
Hur kan vi hjälpa till?
Jag vill även ha F & G. Men Kanban-gränsen hindrar mig. Hmm....
H
I
J
L
K
D
A
Vi behöver inte mer hjälp just nu. Men det här är ett återkommande problem så skriv tester för det så att det undviks i framtiden!


Behov
Utvalt
2
Utveckling
2
Pågående
Klart
Driftas
1
Live!
G
F
H
I
J
L
K
D
C
B
A
Finns det något jag kan göra? Jag är ingen teknikexpert men...
Absolut! Hämta mer kaffe och hjälp oss att inte bli störda.


Några dagar senare...
Behov
Utvalt
2
Utveckling
3
Pågående
Klart
Driftas
1
Live!
M
H
L
J
R
I
G
F
D
K
A
B
C
M eller H. Definitivt M!
Nej, H! Hmmmmm. Nej vänta, M! M eller H eller M eller H?!? Aaaaarrrrgh
LOL
Var det inte bättre förr? Integrationshelvete hela natten? Big-bang-leveranser?

Måste Kanban-tavlan se ut så här?

Nej, tavlan ovan är bara ett exempel!

Det enda som Kanban föreskriver är att arbetsflödet bör vara synligt och att PA-gränser bör sättas. Syftet är att skapa ett jämnt flöde genom systemet och att minimera ledtiden. Du måste regelbundet ta upp frågor som:

Vilka kolumner bör vi ha?
Varje kolumn representerar ett tillstånd i arbetsflödet eller en kö (buffert) mellan två tillstånd. Börja enkelt och lägg till kolumner vid behov.

Vad bör Kanban-gränserna vara?
När Kanban-gränsen för ”din” kolumn nåtts och du inte har något att göra så kan du börja titta ”nerström” (alltså på uppgifter som tornar upp till höger på tavlan) och hjälpa till att åtgärda flaskhalsar. Om det inte finns några flaskhalsar är det en indikation på att Kanban-gränsen kan vara för låg eftersom syftet med gränsen var att minska risken att mata på och skapa flaskhalsar nerströms.

Om du upptäcker att för många uppgifter står stilla under längre tid utan att bli arbetade på är det en indikation på att Kanban-gränsen kan vara för hög.

- För låg Kanban-gräns => inaktiva människor => låg produktivitet
- För hög Kanban-gräns => inaktiva uppgifter => lång ledtid

Hur stränga är Kanban-gränserna?
Vissa lag ser dem hårda (dvs. laget får inte överskrida en gräns). Andra lag ser dem som riktlinjer och diskussionsstartare (dvs. att bryta en gräns är tillåtet men det ska vara ett medvetet beslut med ett konkret syfte). Så, återigen, det är upp till dig. Visst sa jag att Kanban inte var särskilt föreskrivande?

16

Sammanfattning av Scrum vs. Kanban

Likheter

- Båda är Lean och Agila
- Båda använder ”dragplanering” (se kapitel 13).
- Båda begränsar PA.
- Båda använder transparens för att driva processförbättringar
- Båda fokuserar på att leverera levererbar programvara tidigt och ofta.
- Båda baseras på självorganiserande lag
- Båda kräver att arbete bryts ner i bitar
- I båda optimeras leveransplanen kontinuerligt baserat på empiriska data (hastighet / ledtid)

Skillnader

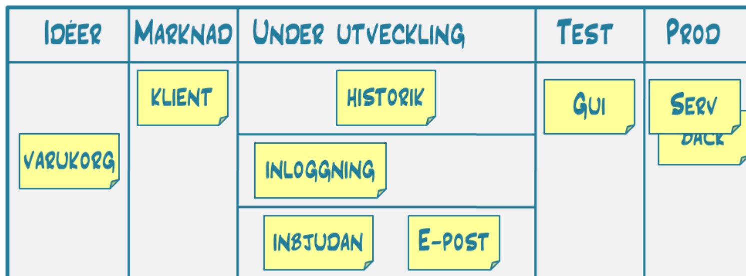
Scrum	**Kanban**
Föreskriver tidsbegränsade iterationer.	**Tidsbegränsade iterationer är valfritt.** Kan ha olika rytmer för planering, leverans och process-förbättring. Kan vara händelse-styrt istället för tidsbegränsat.
Laget förbinder sig till en viss mängd arbete för iterationen.	**Åtagande är valfritt.**
Använder **hastighet** som ett standardmått för planering och processförbättring.	Använder **ledtid** som standardmått för planering och processförbättring.
Föreskriver **tvärfunktionella lag**	Tvärfunktionella lag är valfritt. **Specialistlag tillåtna.**
Uppgifter måste brytas mer så att de kan slutföras inom 1 sprint.	Ingen särskild uppgiftsstorlek föreskrivs.
Föreskriver progressdiagram	Ingen särskild sorts diagram föreskrivs.
PA begränsas indirekt (per sprint)	**PA begränsas explicit** (per tillstånd i arbetsflödet)
Föreskriver estimering	**Estimering är valfritt**
Uppgifter kan läggas till pågående iteration	**Uppgifter kan läggas till så fort det finns tillgänglig kapacitet.**
En **uppgiftslista för sprinten ägs av ett specifikt lag**	En **Kanban-tavla kan delas av flera lag** eller individer.
Föreskriver 3 roller (PÄ/SM/Lag)	**Föreskriver inga roller**
En **Scrum-tavla rensas** mellan varje sprint	En **Kanban-tavla är beständig**
Föreskriver en prioriterad behovslista för produkten.	**Prioritering är valfritt**

Sådär. Det var det, det. Nu vet du vad skillnaderna är.

Men det är inte slut än! Nu är det dags för den bästa delen! Ta på stövlarna för nu är det dags att hoppa ner i skyttegraven med Mattias och se hur det här ter sig i praktiken!

Del II – Fallstudie

Kanban i verkliga livet





Det här är historien om hur vi lärde oss förbättring genom Kanban. När vi började fanns det inte mycket information tillgänglig och Dr Google lämnade oss för en gångs skull tomhänta. Idag utvecklas Kanban framgångsrikt och det finns en växande kunskap tillgänglig. Jag kan varmt rekommendera att ta en titt på David Andersons arbete, exempelvis rörande 'classes of service'. Så här kommer den första (och sista) friskrivningen (jag lovar!). Vilka lösningar du än genomför, se till att de adresserar de specifika problemen. Sådär, klart. Låt oss köra igång. Det här är vår berättelse.

/Mattias Skarin

17

Driftavdelningens natur

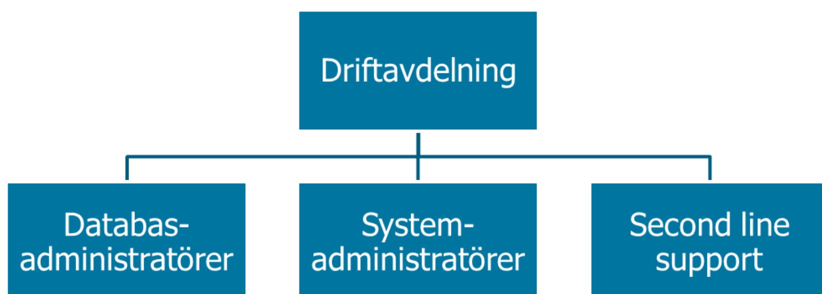
On du någonsin varit på jour 24/7 så har du en god uppfattning om hur det känns att ansvara för en produktionsmiljö. Du förväntas reda ut problem mitt i natten oavsett om du är källan till problemet eller ej. Ingen vet vad felet är, det är därför de ringer dig. Det är en rätt stor utmaning eftersom du inte byggde hårdvaran, drivrutinerna, operativsystemet eller den anpassade programvaran. Dina möjlighter är ofta begränsade till att begränsa effekten, spara bevis för att återskapa problemet och att vänta på den person som är ansvarig för att orsakat besvären så att denna kan reproducera och lösa problemet som du just bevittnat.

För driftavdelningen är hastighet och nogrannhet i både svarstider och problemlösning viktiga.

18

Drivkraft till förändring

2008 genomförde en av våra kunder, ett skandinaviskt spelutvecklingsföretag, en mängd processförbättringar. En av dessa förbättringar innebar uppskalning av Scrum till hela utvecklingsavdelningen samt en successiv eliminering av hinder som förhindrade utvecklingsteamen att leverera programvara. När programvaran började flöda och kapaciteten ökade, ökade också trycket på driftavdelning. Tidigare hade IT-teknikerna mest stått och sett på från håll – nu blev de alltmer involverade som en aktiv part i utvecklingsprocessen.



Figur 1. Företagets driftavdelning omfattade tre team: databasadministratörer, systemadministratörer och second line support.

En konsekvens av detta var att det inte räckte med att hjälpa utvecklingsteamen. Om man fortsatte att uteslutande fokusera på att hjälpa utvecklingsteamen skulle det orsaka förseningar i det kritiska förbättringsarbetet av infrastrukturesom leddes av driftavdelningen. Det behövdes förbättringar på båda områden.

Dessutom ledde framstegen på utvecklingssidan till att cheferna efterfrågades alltmer för hjälp med analys och återkoppling av idéer i tidiga projektstadier. Det betydde i sin tur att de fick mindre tid över till löpande prioritering och problemlösning. Ledningsgruppen insåg att de måste agera innan situationen blev ohanterlig.

19

Var börjar vi?

En bra början var att fråga utvecklingsteamet, dvs. driftavdelningens kund.

Utvecklings syn på driftavdelningen

Jag ställde frågan “vilka är de tre viktigaste sakerna du kommer att tänka på vad gäller ’IT-drift’? De vanligaste svaren var:

“Varierande kompetens”

“Mycket kompetenta i IT-infrastruktur”

“De vill hjälpa till men faktum är att det är svårt att få hjälp.”

“Projekten tar för lång tid”

“Supportsystemet suger”

“Vad håller de på med?”

“Det behövs många e-mail för enkla saker”

“Svåra att kontakta”

I korthet var detta utvecklingsteamets syn på driftavdelningen. Låt oss nu jämföra med vad driftavdelningen tyckte om utveckling…

Driftavdelningens syn på utveckling

”Vi”
(driftavdelningen)





”Dom”
(utveckling)



“Varför använder ni inte fördelarna med den befintliga plattformen?”

“Låt oss göra releasehanteringen mindre tungrodd!”

“Vi lider av er dåliga kvalitet!”

“*De* borde ändra på sig” var ett återkommande tema i argumenten från båda sidor. För att lösa de gemensamma problemen skulle vi uppenbarligen behöva ändra det synsättet. På pluskontot fanns gott förtroende för kärnkompetensen (”mycket kompetenta i IT-infrastruktur”) vilket fick mig att tro att ”vi och dom”-mentaliteten skulle kunna rättas till om vi bara skapade rätt förhållanden.

20

Sätta igång

Så vi behövde sätta igång. Men var skulle vi börja?

Jag hade en bakgrund som programmerare så jag visste verkligen lite om IT-drift. Jag tänkte inte ”storma in och ändra på saker”. Jag behövde en mindre konfronterande ansats som alltjämt skulle lära oss de viktiga sakerna, förkasta de irrelevanta sakerna och vara lätt att lära.

Kandidaterna var:

1. Scrum – detta fungerade bra i utvecklingslagen
2. Kanban – nytt och oprövat men med en bra passform till Lean-principerna som vi saknade.

Vid diskussion med cheferna verkade Kanban och Lean-principerna kunna tackla de problem som vi försökte adressera. Från deras perspektiv skulle inte sprintar passa särskilt bra eftersom man omprioriterade dagligen. Därför var Kanban en logisk startpunkt även om det var nytt för oss alla.

21

Köra igång teamen

Hur skulle vi köra igång teamen? Det fanns ingen tillgänglig handbok i hur man kör igång. Om vi gjorde fel skulle det innebära stora risker. Förutom att gå miste om förbättringarna skulle vi hantera en produktionsplattform med synnerligen specialiserade och skickliga utvecklare som var svåra att ersätta. Att fjärma dem skulle vara en dååålig idé.

- Skulle vi bara sätta igång och hantera konsekvenserna allt eftersom de uppstod?
- Eller skulle vi köra en workshop först?

För oss var det uppenbart – vi skulle först köra en workshop. Men hur? Det var en utmaning att få hela driftavdelningen att delta i en workshop (vem svarar då om någon ringer?). Till slut bestämde vi oss för att genomföra en halvdags-workshop och att hålla den enkel och övningsbaserad.

Workshopen

En av fördelarna med en workshop var att det tidigt skulle blottlägga våra problem. Det bidrog också till att skapa en förtroendeingivande miljö i vilken implikationer kunde diskuteras direkt med deltagarna. Ärligt talat – alla var inte överdrivet entusiastiska över att ändra arbetssätt. De flesta var dock öppna för att prova. Så vi genomförde en workshop där vi demonstrerade de viktigaste principerna och körde en nerskalad Kanban-simulering.

Lära några grundläggande principer

- Begränsa arbete efter kapacitet.
- Batchstorlek vs. cykeltid.
- Pågående arbete vs. genomströmning.
- Restriktionsteori.

Kanban-demo

- 3 "arbetstyper";
  besvara frågor,
  bygg en Lego-bil,
  Designa och bygg ett hus.
- 3 iterationer
  Mät hastighet per arbetstyp.
  Experimentera, justera PA.
- Avrapportera.

I slutet av workshopen gjorde vi en "femfinger"-röstning för att kontrollera om teamen var villiga att prova det här i verkligheten. Inga invändningar framkom så vi beslöt att sätta igång. *(Femfingerröstning är en konsensusskapande metod där varje deltagare indikerar sin enighet på en skala från 1 till 5 fingrar. 5 betyder störst enighet, 1 betyder mest oenig, 3 betyder enighet med reservation. Konsensus har uppnåtts när varje deltagare i gruppen håller upp minst 3 fingrar.)*

22

Adressera intressenter

Personer beroende av driftavdelningens jobb (även kallat ”stekhållare”) skulle sannolikt bli påverkade av introduktionen av Kanban. Visserligen var förändringarna till det bättre – det skulle innebära att laget började säga ”nej” till arbete som de inte skulle kunna slutföra, stå upp för kvalitet och ta bort lågprioriterade uppgifter från lagets uppgiftslista. Trots detta är det alltid en bra idé att först ha en dialog.

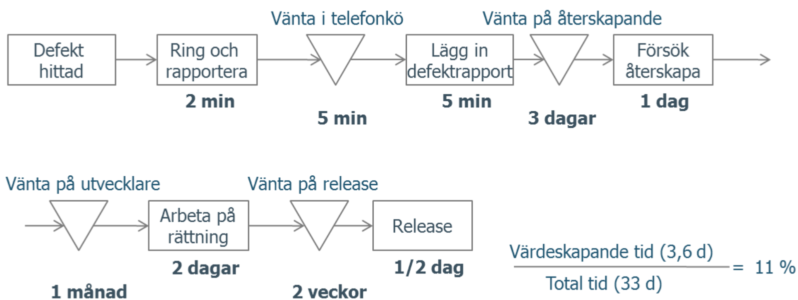
De närmsta intressenterna var “first-line support” och avdelningscheferna. Eftersom de hade deltagit i workshopen så var de reda positivt inställda till att gå vidare. Samma sak gällde utvecklingsteamen (som mer eller mindre ändå förväntade sig förbättringar). Men för ett team som support, var saken en annan. Deras mest signifikanta problem var att de var överbelastade. Dessutom hanterade de alla kundärenden och företaget hade åtagit sig att svara på alla ärenden. Detta skulle sannolikt förändras om vi introducerade Kanban och började tvinga fram begränsningar på pågående arbete – PA (work in progress, ”WIP” på engelska *förf. anm.*).

Så vi hade ett möte med de viktigaste intressenterna där vi presenterade vår intention, förväntade fördelar och möjliga konsekvenser. Till min lättnad togs våra idéer generellt sett väl emot, ibland med kommentaren ”utmärkt om vi äntligen kan hantera de här sakerna en gång för alla”.

23

Hur vi skapade den första tavlan

En bra start för att skapa en Kanban-tavla är att kartlägga värdeflödet. Det är i grund och botten en visualisering av värdekedjan och ger en insikt i arbetstillstånd, flöde och tid genom systemet (cykeltid).



Men vi började mycket enklare: ett förslag ritat på papper tillsammans med chefen. Granskad ett antal gånger och sedan körde vi igång. Frågor som togs upp under denna fas var bland andra:

- Vilka typer av arbete har vi?
- Vem hanterar det?
- Ska vi dela ansvar för olika arbetstyper?
- Hur hanterar vi delat ansvar när vi har specialiserade kunskaper?

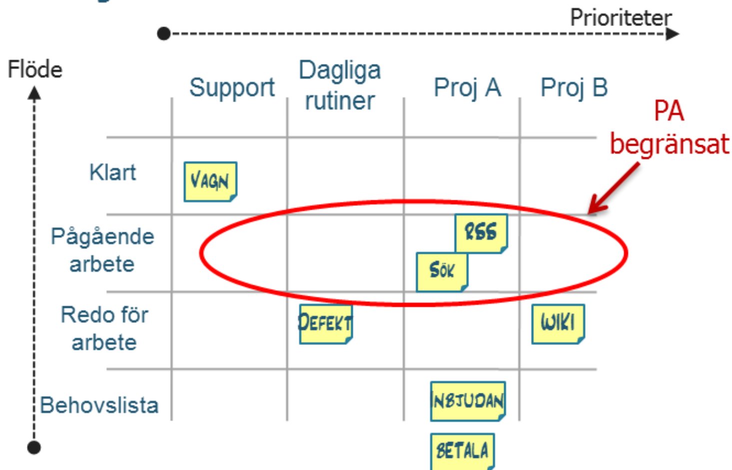
Eftersom de olika arbetstyperna hade överenskomna servicenivåer (SLA) kändes det naturligt att låta varje lag styra utformningen av sin egen tavla. De tog fram kolumner och rader själva.

Nästa stora beslut var huruvida ansvar skulle delas för olika arbetstyper. ”Skulle vi låta en fast del av teamet hantera direkta frågor (reaktivt arbete) och låta resten av laget fokusera på projekten (proaktivt arbete)?” Vi bestämde oss först för att prova delat ansvar. En nyckelorsak till det var att vi hade identifierat att självorganiserande lag och fortsatt lärande och

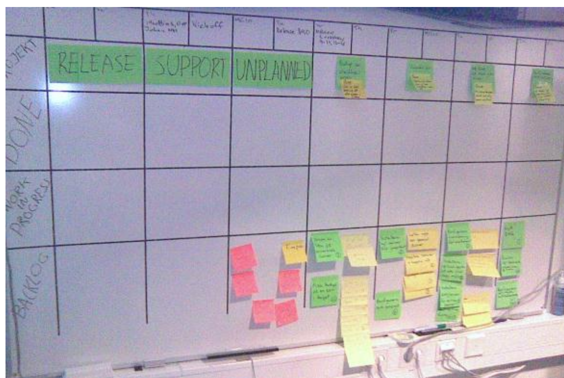
kunskapsöverföring bland lagdeltagarna var nödvändigt för att säkerställa tillväxt. Nackdelen med beslutet var de potentiella avbrott alla kunde drabbas av men det var den bästa lösningen vi kunde komma fram till från början. En liten notering: när vi körde workshopen så självorganiserade sig faktiskt lagen kring detta problem. De lät en person hantera de omedelbara förfrågningarna och resten hantera de större ärendena.

Den första Kanban-modellen

Nedan visas den grundläggande modellen som vi använde för Kanban. Notera att laget valde att låta uppgifter flöda uppåt (som vattenbubblor) istället för att följa den mer typiska modellen med flöde från vänster till höger.



Figur 2. Den första modellen av Kanban-tavlan. Prioriteter går från vänster till höger, flödet går uppåt. PA räknas som det totala antalet uppgifter i ”Pågående arbete”-raden (inringat). Modellen är inspirerad av erfarenheter som rapporterats av Linda Cook.



Figur 3. Den första Kanban-tavlan för systemadministrationsteamet.

Rader

Arbetsflödestillstånd	Så definierade vi det
Behovslista	Användarberättelser (”user stories”) som chefen beslutade att man behöver.
Redo för arbete	Användarberättelser som är estimerade och nedbrutna till uppgifter på vardera max 8 timmar.
Pågående arbete	Raden med PA-gränsen. Vi började med en gräns på 2 x lagstorleken – 1 (-1 är för samarbete). Ett 5-personerslag har alltså en PA-gräns på 7.
Klart	Körbart av slutanvändaren.

Kolumner

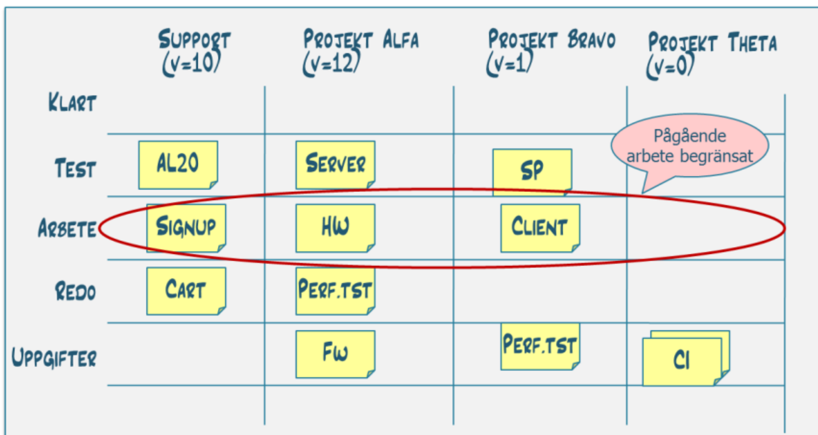
Arbetstyp	Så definierade vi det
Leverans (Release)	Hjälpa utvecklingsteam att leverera programvara.
Support	Mindre förfrågningar från andra lag.
Oplanerat (Unplanned)	Oväntat arbete som måste göras men som inte har en tydlig ägare. Exempelvis mindre förbättringar I infrastruktur.
Project A	Större drifthanteringsprojekt. Exempelvis att byta hårdvaran i en stagingmiljö.
Project B	Ett annat stort projekt.

Alla Kanban-tavlor såg inte likadana ut. Vi började med en enkel skiss som utvecklades allt eftersom.

24

Att sätta den första gränsen för pågående arbete (PA)

Vår första gräns för produkter i arbete var rätt generös. Vi resonerade att genom att visualisera flödet så skulle vi se och uppleva vad som pågick samt att det var osannolikt att vi skulle lyckas gissa den bästa gränsen från början. Med tiden justerade vi PA-gränsen så fort vi fann en god anledning att göra det. Den första PA-gränsen sattes till 2n-1 (n=antal lagspelare, -1 för att uppmuntra samarbeta). Varför? Enkelt. Vi hade ingen bättre idé ☺. Dessutom verkade det okontroversiellt för att komma igång. Formeln erbjöd en enkel och logisk förklaring till alla som försökte tvinga på laget mer arbete: ”...så anta att varje teammedlem kan arbeta på högst två saker samtidigt, en aktiv och en väntande, hur kan vi förvänta oss att de ska kunna ta sig an *mera*?”. När jag ser tillbaka på inser jag att vilken generös gräns som helst skulle gått bra att börja med. Genom att bevaka Kanban-tavlan är det enkelt att hitta de rätta gränserna allt eftersom.



Figur 4. Hur vi applicerade gränserna för pågående arbete för DBA- och systemadministrations-lagen. En gräns för varje arbetstyp.

En observation vi gjorde var att det var meningslöst att definiera PA-gränser i användarberättelsepoäng (”story points”). Det var helt enkelt alldeles för svårt att hålla koll på. Den enda gränsen som var lätt nog att hålla koll på var helt enkelt att räkna antalet uppgifter (= parallella uppgifter).

För supportteamet använde vi PA-gränser definierat per arbetstyp (kolumn) eftersom vi behövde snabbare reaktion om gränsen överträddes.

25

Hur vi lärde oss respektera produkter i arbete

I teorin låter det enkelt att respektera PA-gränserna men i praktiken är det en utmaning. Det innebär nämligen att i något skede säga ”nej”. Vi har prövat olika ansatser för att hantera detta.

Diskutera framför tavlan

Om en överträdelse upptäcktes förde vi samman intressenterna framför tavlan och frågade vad de ville uppnå. I början berodde överträdelser oftast på oerfarenhet. I vissa fall identifierade vi olika syn på prioriteringen där ett typiskt fall utgjordes av lagspelare som var specialist inom ett specifikt område. Det var de enda gångerna som vi upplevde någon friktion. I merparten av fallen reddes problemen ut där och då genom att diskutera framför tavlan.

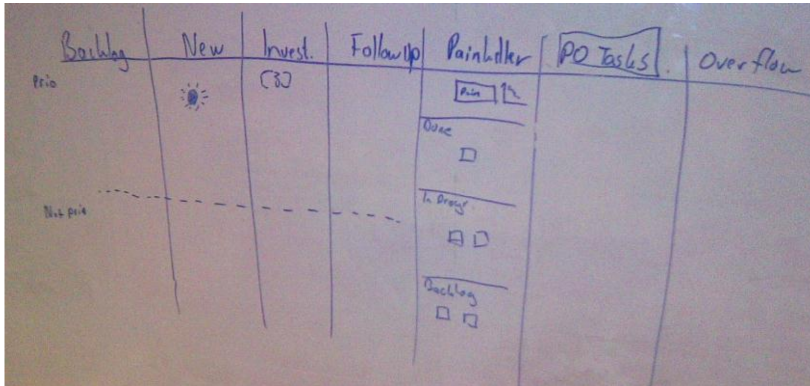
Skapandet av en översvämningssektion

I fallet med supportteamet hade vi ett dilemma med att införa PA-gränser. Enligt reglering på marknaden var de tvunga att lova svara på alla frågor, så det var svårt att säga nej till vissa frågor. Det var helt enkelt för konfronterande att säga ”nej”. Vi löste det genom att införa en översvämningssektion som trädde i kraft så snart PA-gränsen överträddes. Två regler gällde för att få flytta uppgifter till översvämningssektionen:

1. En kontakt tas med behovsställaren. Vi informerade om att vi inte glömt bort ärendet men att vi inte kunda ta det just nu, so fort det fanns ledig tid sklle vi göra det. Vi föreslog även att om det fanns en alternativ lösning vore det klokt att bruka den

2. Om vi tar bort uppgiften så blir du informerad om det.

Efter två veckor var det uppenbart att de översvämmade uppgifterna troligtvis aldrig skulle hanteras så med stöd från chefen plockade vi bort dem från tavlan.

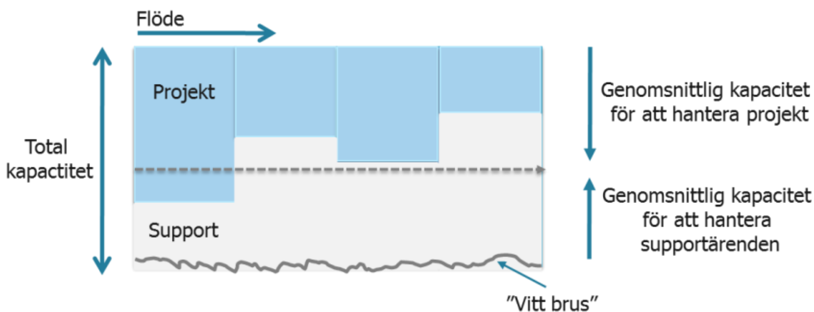


Figur 5. En skiss over Kanban-tavlan för supportlaget. Översvämningssektionen är längst till höger.

26

Vilka uppgifter hamnar på tavlan?

Vi bestämde tidigt att inte ta alla typer av arbete som teamet utför skulle hamna på tavlan. Att bevaka saker som telefonsamtal eller hämta kaffe skulle göra Kanban till ett administrativt monster. Vi var här för att *lösa* problem, inte skapa dem ☺. Så vi bestämde att endast sätta upp uppgifter med en storlek >1 timme på tablan och allt därunder betraktade vi som ”vitt brus”. 1-timmesgränsen fungerade faktiskt rätt bra och var en av de saker som förblev oförändrade. (Vi var tvungna att revidera våra antaganden om vilken påverkan bakgrundsbruset skulle ha men mer om det senare).



Figur 6. Vi började med att anta att den totala kapaciteten främst konsumerades av två arbetstyper, större (projekt) och mindre (support). Genom att hålla reda på hastigheten för projekt kunde vi få en indikation på leveransdatum om det behövdes. ”Vitt brus” (mindre support <1 timme, möten, hämta kaffe, hjälpa kolleger) förväntade vi alltid skulle finnas.

27

Hur estimera?

Det finns flera alternativ och det finns sannerligen mer än ett svar:

- Estimera regelbundet
- Estimera när det finns ett behov
- Använd ideala dagar/användarberättelsepoäng som estimat
- Estimat är osäkra, använd något enkelt, tex. T-tröjestorlekar (Liten, Medium, Stor)
- Estimera inte eller estimera bara när det finns en fördröjningskostnad som motiverar det.

Något influerade av Scrum (eftersom det trots allt var därifrån vi kom) bestämde vi oss att börja med historiepoäng. Men i praktiken behandlade teamet användarberättelsepoäng som om de vore ekvivalenta med mantimmar (det kändes mer naturligt). I början estimerade vi alla användarberättelser. Med tiden lärde sig cheferna att om de höll nere antalet pågående projekt så skulle de inte tvinga intressenterna vänta. De lärde sig också att om det uppstod en plötslig förändring så kunde de prioritera om och adressera problemet.

När vi insåg att vi oftast levererade våra delar av projekt innan andra behövde dem minskade behovet av estimerimg. Detta ledde till att chefer slutade fråga efter tidiga estimat. De gjorde endast det om de var oroliga för att de skulle tvinga någon vänta längre än väntat.

I ett tidigt skede, stressad av ett telefonsamtal, lovade en chef projektleverans ”i slutet av denna veckan”. Då projektet fanns på Kanban-tavlan var det enkelt att estimera framstegen (genom att räkna % fördigt arbete) och dra slutsatsen att ungefär 25% var klart efter en vecka). Alltså krävdes ytterligare tre veckor. Konfronterad med fakta ändrade chefen projektets prioritet, stoppade parallellt arbete och gjorde leveransen möjlig. Kolla alltid med tavlan. ☺

Vad betyder estimerad storlek? Ledtid eller arbetstid?

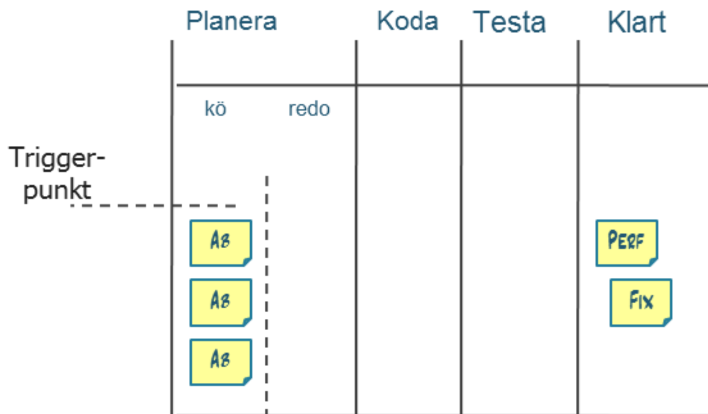
Våra användarberättelsepoäng representerade arbetstid, dvs. hur många timmar oavbrutet arbete som vi förväntade oss att användarberättelsen skulle ta. Inte ledtid (eller kalendertid; eller antal timmars väntetid). Genom att mäta antalet användarberättelsepoäng som nådde ”klart” varje vecka (hastighet) så kunde vi härleda ledtiden.

Varje ny användarberättelse estimerade vi endast en gång. Vi reviderade inte estimaten under tiden vi arbetade med uppgiften. Detta gjorde att vi kunde minimera tiden som laget spenderade på estimering.

28

Så hur arbetade vi egentligen?

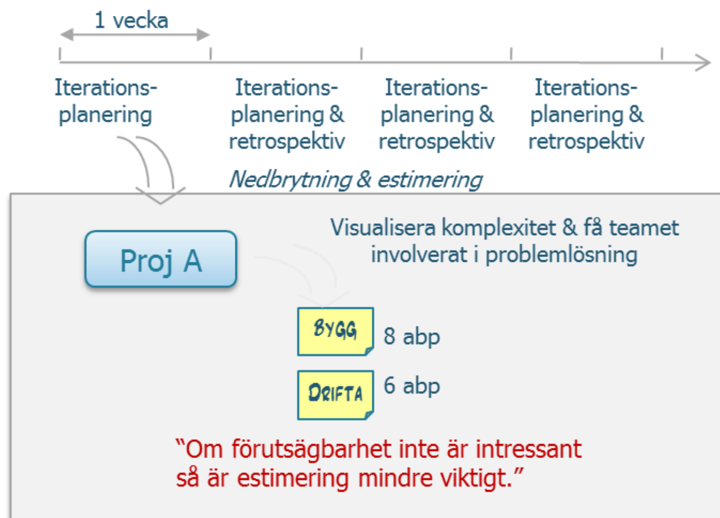
Kanban har verkligen få restriktioner – du kan arbeta på alla möjliga sätt. Du kan låta laget arbeta med tidsbestämda aktiviteter eller så kan du välja att utföra aktiviteter när tillräcklig momentum har byggts upp för att motivera det.



Figur 7 När tre uppgifter har inommit uppgiftslistan triggas planering/estimering.

Vi valde att schemalägga två återkommande händelser:

- Dagligt ståuppmöte – med laget placerat framför tavlan för att lyfta problem och för att hjälpa till att skapa en gemensam syn på de andra lagspelarnas uppgifter.
- Veckovis iterationsplanering – med syfte att planera och kontinuerligt förbättra.



Detta fungerade bra hos oss.

Dagligt ståuppmöte

Det dagliga ståuppmötet påminde om ett dagligt Scrum-möte. Detta hölls efter det dagliga ”Scrum of Scrums”-mötet med deltagare från alla funktioner (utveckling, test, driftsättning). ”Scrum of Scrums” gav viktig information till Kanban-teamen, t.ex. vilka problem som skulle tas om hand först och vilka utvecklingsteam som för närvarande hade mest problem. I början deltog cheferna ofta på dessa dagliga ståuppmöten och föreslog lösningar och prioriterade beslut. Med tiden, allt eftersom deltagarna blev bättre på att självorganisera, deltog cheferna allt mer sällan (men var inte långt bort om de behövdes).

Iterationsplanering

En gång i veckan genomfördes iterationsplanering. Vi höll det varje vecka vid en bestämd tidpunkt eftersom vi insåg att om vi inte planerade in det skulle tiden förbrukas av andra prioriteringar ☺. Vi behövde också bättre kommunikation och kunskapsöverföring. En typisk agenda var:

- Uppdatera diagram och tavla. (Färdiga projekt flyttades till en “färdigställandets vägg”.
- Titta tillbaka på förra veckan. Vad hände? Varför då? Vad kan vi göra för att förbättra det?

- Justering av PA-gränser (om nödvändigt)
- Nedbrytning av uppgifter samt estimer för nytt projekt (vid behov)

I grunden var iterationsplaneringen en kombinerad puls av estimering och ständiga förbättringar. Små och medelstora frågor löstes på plats med stöd av linjecheferna. En svårare prövning var att hålla greppet om komplexa förbättringar. För att hantera detta införde vi möjligheten för temen att tilldela upp till 2 ”laghinder” till sina chefer.

Reglerna var som följer:

1. En chef kan arbeta på två problemluckor vid varje givet tillfälle.
2. Om båda är fulltecknade kan du bara lägga till en ny om du tar bort en mindre viktig.
3. Teamet bestämmer när ett problem är löst.



Detta var en positiv förändring. Plötsligt kunde teamen se att cheferna aktivt hjälpte dem även med de svåra problemen. De kunde peka på hindren och fråga ”hur går det?” De skulle inte bli bortglömda eller överkörda av en ny prioriteringsstrategi.

Ett exempel på ett allvarligt hinder var att när driftteamen misstänkte att det fanns ett produktionsproblemfick de inte den hjälp de behövde från utvecklare. De behövde den hjälpen för att förstå vilket delsystem som kunde orsaka problemet. Men eftersom utvecklarna var upptagna med att utveckla nya funktioner i sprintarna började problemen staplas på hög. Föga förvånande så upplevde driftteamen att utvecklarna inte brydde sig tillräckligt om kvalitet.

När detta hinder blottlades eskalerades det först till linjechefen och sedan vidare till avdelningschefen. Han bokade in ett möte med utvecklingschefen. I de diskussioner som följde beslöt cheferna att sätta kvaliteten först. De mejslade fram en ”round-robin”-lösning - för varje sprint skulle ett utvecklingslag hålla jour och vara omedelbart tillgängligt för att hjälpa driftavdelningen. En stor skillnad var också att

utvecklingssidan från och med nu sa att de skulle lita på driften om de ansåg att det var ett allvarligt produktionsproblem. Tills nu hade driften varit tvungen att bevisa att det var ett allvarligt problem för att få deras hjälp.

Efter att ha säkrat stöd hos sina chefer lämnade utvecklingschefen över en lista på kontaktpersoner till supportlagen. Det tog fem sekunder från att åtgärden annonserades till teamen tills att de satte den på prov. Men det var på riktigt den här gången. Rätt förberedelser hade blivit gjorda och hindret var överkommet – till stor lättnad för driftteamen.

29

Hur vi fann ett planeringskoncept som fungerade

En berättelse

Jag minns en vändpunkt för ett av teamen. Jag satt med dem i deras andra estimeringsmöte. Laget hade kört fast i ett projekt som de inte viste hur man skulle estimera. Det fanns för många okända parametrar och hela estimeringsmötet avstannade. Istället för att gå in och ta över så bad jag dem att förfina processen för att hitta en bättre lösning. Med sin chef som ledsagare tog de sig an uppgiften och började skapa en egen lösning. Denna händelse var en viktigt vändpunkt – en viktig ”seger” – från vilken de började utvecklas till ett lag med självförtroende. Efter detta började de utveckls så snabbt att vi var tvunga att stå åt sidan.

Två månader senare kom deras chef fram till mig efter ett retrospektivmöte. ”Jag har ett problem”, sa han och pekade på Kanban-tavlan. ”Vi har inga egentliga problem. Vad ska vi göra?”

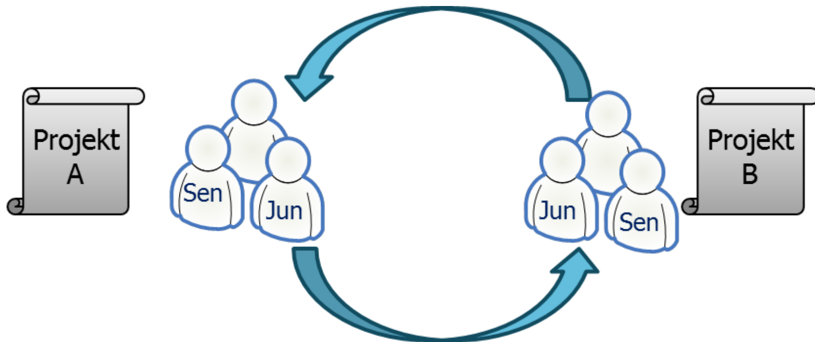
Återupptäcka planering

Planeringspoker på estimeringsmöten som omfattade alla teammedlemmar fungerade inte bra för något av teamen. Några orsaker:

1. Kunskapen var ojämn inom laget.
2. Vanligtvis pratade endast en person.
3. Lagdeltagarna ville ta tag i brådskande ärenden som de hade lämnat på sitt arbetsbord.

Men genom lite experimenterande kom teamen, oberoende av varandra, fram till olika estimeringsprocesser. Varje process fungerade bra för respektive team.

Tillvägagångssätt 1 – Växla och granska



- För varje projekt/berättelse: tilldela ett ”senior + junior”-par som får estimera (dvs. en person som känner väl till den aktuella berättelsen och en person som inte gör det). Detta hjälper till att dela kunskap.

- De återstående lagdeltagarna väljer vilken berättelse som de vill estimera (men med en gräns på fyra personer per berättelse för att få effektiva diskussioner)

- Varje estimeringsgrupp bryter ner sin berättelse till uppgifter och, om det behövs, estimerar den.

- Slutligen byter grupperna berättelser och granskar varandras resultat (en person per lag ”stannar kvar” och förklarar för granskarna hur man resonerat)

- Klart!

Hela iterationsplaneringsmötet tog typiskt omkring 45 minuter och energinivåerna var kvar på en hög nivå mötet igenom. Vanligtvis gjordes 1 till 2 justeringar när berättelserna skickades runt och granskades av nya ögon.

Tillvägagångssätt 2 – senior genomgång, sedan estimering

Två seniora medlemmar började med att göra en grov genomgång av berättelsen/projektet före planering. Den första frågan de ställde sig var ”förstår vi tillräckligt om problemet för att skapa en lösning?” Om svaret var nej returnerade de projektet till uppdragsgivaren. Om svaret var ja skapade de en grov lösning, i stort vilka teknologier som skulle användas. Teamet tog sedan över och bröt ned projektet till uppgifter med den grova lösningen som utgångspunkt.



Figur 8. Nedbrytning till uppgifter med ett annat lag som granskar på iterationsplaneringen.

30

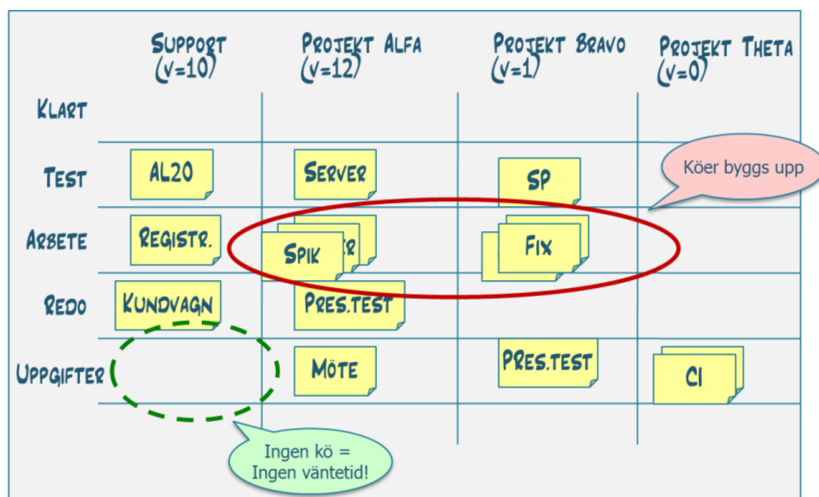
Vad ska mätas?

Det finns mycket som kan mätas – ledtid (tiden det tar från det att ett behov upptäcks tills det att behovet är uppfyllt), hastighet, köer, progress... Den viktiga frågan är: vilka mått kan *användas* för att förbättra processen? Mitt råd är att experimentera och se vad som funkar för er. Vi lärde oss att progressdiagram var onödigt för projekt som är kortare än 4 veckor. Den övergripande framåtskridandet kan enkelt fastställas genom att titta på Kanban-tavlan (hur många berättelser var det i uppgiftslistan och hur många blev klara).

Potentiellt mått	**För**	**Emot**
Ledtid	Lätt att mäta. Behöver inga estimat. Börjar och slutar med kunden.	Väger inte in storlek.
Total hastighet (sammanslagning av alla arbetstyper)	En grov men enkel indikator för förbättring eller variation.	Hjälper inte för att prognostisera leveransdaum för specifika arbetstyper.
Hastighet per arbetstyp	Mer precist än totalt hastighet.	För att vara användbart måste det starta från kundbehovet och följa hela vägen till leverans. Tar lite längre tid att följa jämför med total hastighet.
Kölängder	En snabb indikator av behovsfluktuationer. Lätt att visualisera.	Säger inget om orsaken är ojämnt inflöde av behov eller ojämn kapacitet. En kölängd på noll kan

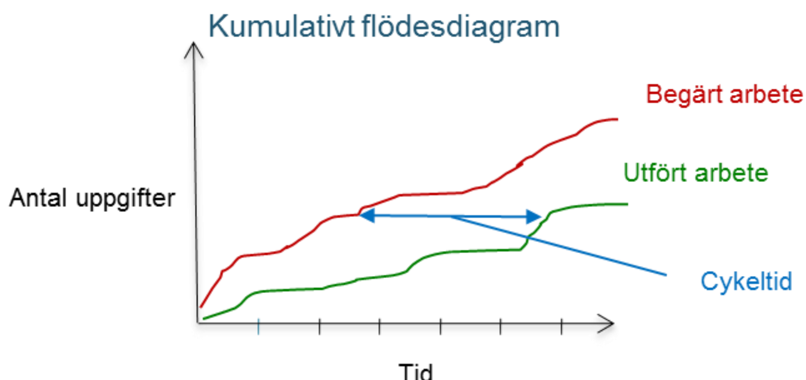
		indikera överkapacitet.

Vi började med att mäta "hastighet per arbetstyp" och "kölängder". Hastighet per arbetstyp är enkelt att mäta och är användbart. Kölängder är bra huvudindikatorer eftersom de kan upptäckas omedelbart (när du vet var du ska hitta dem).



Figur 9. Flakshalsar och möjligheter. Den röda cirkeln visar hur köer har byggts upp och avslöjar flaskhalsar i test. *Frånvaron* av kö i uppgiftslistan i supportkolumnen indikerar att det inte är någon väntetid för nytt supportarbete. Detta är ett gott tecken på hög kundservicenivå.

Vi använde inte kumulativa flödesdiagram men det kunde varit intressant.



Vi använde inte kumulativa flödesdiagram eftersom Kanban-tavlan och hastighetsdiagrammet gav oss tillräckligt med information. Åtminstone i tidiga mognadsfaser. Flaskhalsar, ojämnhet och för mycket arbete kan kunde ändå enkelt identifieras och att lösa det höll oss sysselsatta de första sex månaderna.

31

Hur saker och ting började förändras

Tre månader efter att Kanban införts gav ledningen system-administrationsteamet utmärkelsen “bäst presterande lag” på IT-avdelningen. Samtidigt röstades samma team fram som en av de tre mest ”positiva erfarenheterna” i företagets retrospektiv. Företagets retrospektiv är en företagsomfattande aktivitet som inträffar var 6:e vecka och detta var första gången som ett *team* hamnade på topp-3-listan! Och bara 3 månader tidigare besvärdes dessa lag av flaskhalsar som de flesta klagade på.

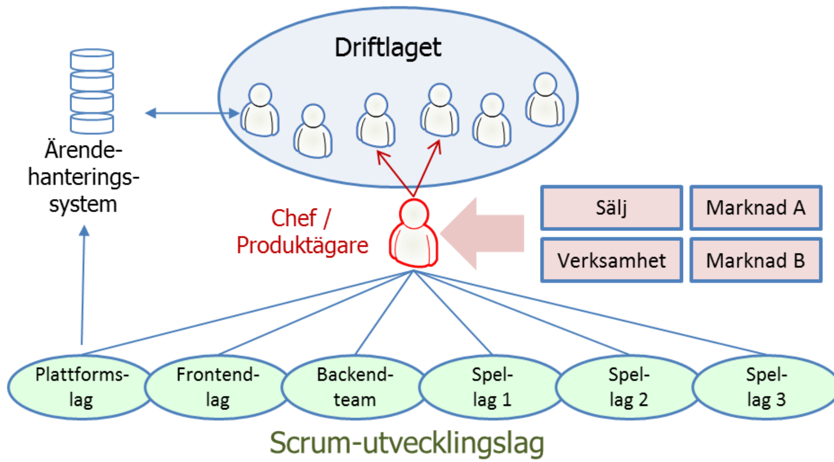
Tjänstekvaliteten hade ökat. Så hur gick det till?

Det väsentliga ögonblicket var när alla började dra tillsammans. Cheferna gav ett tydligt fokus och skyddade teamen från arbete som inte var deras och teamen tog ansvar för kvalitet och deadlines. Det tog ungefär tre till fyra månader innan detta inträffade men efter det så flöt det på. Alla problem försvann inte (då skulle vi alla vara utan arbete, eller hur? ☺) men vi ställdes inför nya utmaningar som ”hur håller vi laget motiverat för fortsatta förbättringar (när inte de själva längre är flaskhalsen)”.

En viktigt bit i självorganiseringspusslet var införandet av konceptet med ”en driftskontakt per utvecklingsteam”. Det innebar att varje utvecklingsteam fick sin egen supportkontakt på driftsavdelningen. Kanban gjorde detta möjligt genom att tillåta teammedlemmarna att självorganisera arbetet, undvika övertid och möjliggöra ständig förbättring. Förut tog någon person, slumpmässigt vem, en uppgift från kön, löste det efter bästa förmåga och fortsatte sedan med nästa uppgift. Kommunikationsbrister innebar att man tvingades börja om från början med ett nytt loggat ärende. När ett-till-ett-konceptet infördes gavs supportlaget plötsligt möjligheten att respondera snabbt när dåligt data eller kvalitetsproblem hotade systemet.

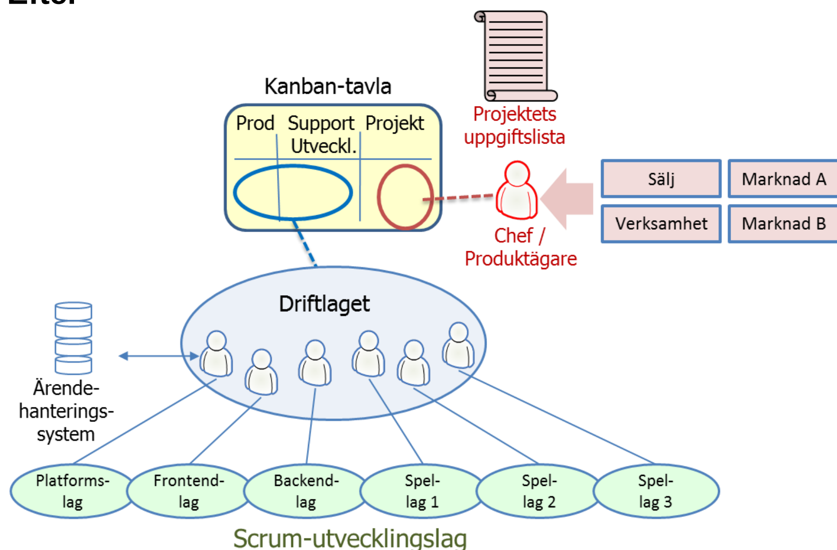
Snabbt utvecklades egenanpassade kommunikationsprotokoll, tex. Började driftspersonalen använda chat för att prata med utvecklare som de kände väl, e-post för de som skrev bättre än de pratade och telefon om det var det snabbaste sättet att lösa problemet ☺.

Före



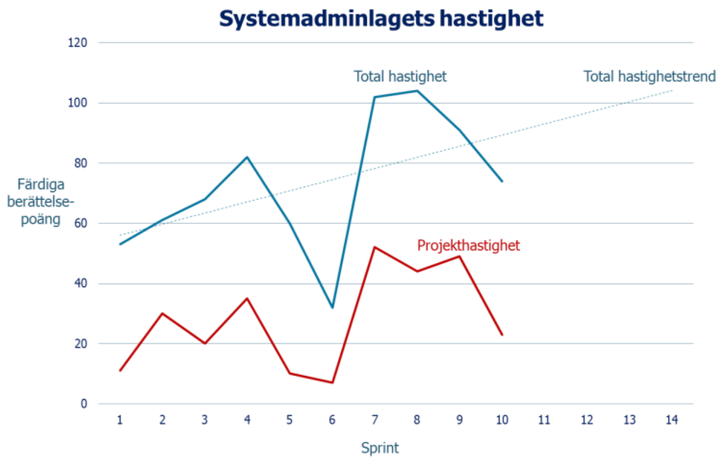
Figur 10. Före: Första linjens chef är huvudkontakten för lagen. Allt viktigt som måste göras går genom honom. Små saker, vanligtvis utvecklarnas problem, tas emot via ett ärendehanteringssystem. Få direkta person-till-person-interaktioner.

Efter



Figur 11. Efter: "en driftskontakt per utvecklingsteam" infördes. Utvecklingslagen pratar direct med en utsedd kontaktperson på driftsavdelningen. Många interaktioner person-till-person. Driftslagets spelare självorganiserar sitt arbete med hjälp av Kanban-tavlan. Chefen byter fokus till att prioritera stora projekt och att ge stöd när svåra problem uppstår.

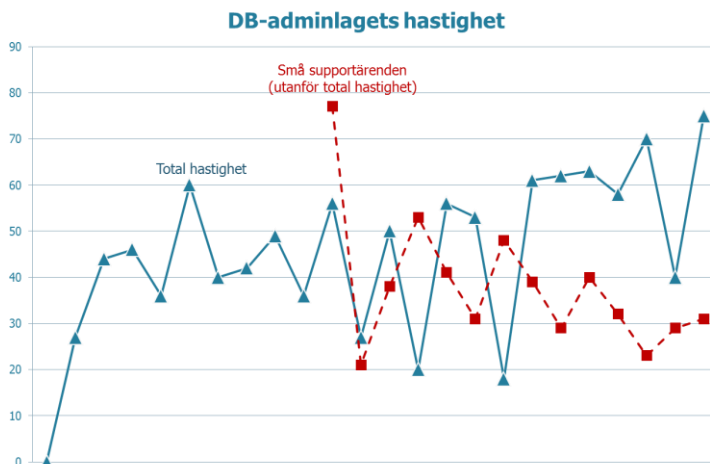
Så hur påverkades teamens prestationer?



Figur 12. Total hastighet och projekthastiget, mätt i antalet färdiga berättelsepoäng per vecka. Totalen är summan över alla kolumner. Projekthastigheten representerar den del som tillägnades ”projekt” (större arbetsuppgifer som att uppgradera en hårdvaruplattform). De två dalarna sammanfaller med 1) en vecka då nästan alla teammedlemmar reste och 2) en stor leverans från utveckling.

Således visade laget en allmänt positiv trend. Samtidigt hade laget investerat i kunskapsdelning och teamarbete.

Och när vi ändå håller på, låt oss kolla på DB-adminlagets prestation.



Figur 13. Total hastighet och små supportuppgifter. Dalarna i mitten sammanfaller med julledigheten.

Trenden för total hastighet är uppåtgående även om variansen är betydande. Variansens storlek inspirerade teamet att övervaka antalet små supportärenden (uppgifter som är för små för att kvalificera sig till Kanban-tavlan). Som du ser visar grafen en tydlig omvänd korrelation mellan antalet små supportärenden och den totala hastigheten.

Supportlaget började med Kanban senare än de andra två lagen så vi har ännu inte särskilt mycket pålitlig data.

Växande mognad

När vi började var det lätt att hitta problemområden. Men att identifiera den största möjligheten till förbättring var svårt. Kanbantavlan gav en mycket stor transparans. Det var inte bara enklare att identifiera problem utan viktiga frågor lyftes om flöde, varians och köer. Vi började använda köer som ett verktyg för att identifiera problem. Fyra månader efter att de börjat med Kanban började cheferna jaga orsakerna till varianser som påverkade lagen negativt.

Allt eftersom teamen utvecklades från individer till självorganiserade enheter upptäckte cheferna att de stod inför en helt ny typ av utmaningar i sitt ledarskap. De tvingades allt mer att hantera personalfrågor – hantera klagomål, definiera gemensamma mål, lösa konflikter och förhandla överrenskommelser. Inte en helt smärtfri förändring – de anmärkte öppet att det tog mycket krävdes både skicklighet och energi för att lära sig det. Men antog utmaningen och blev därmed också bättre ledare.

32

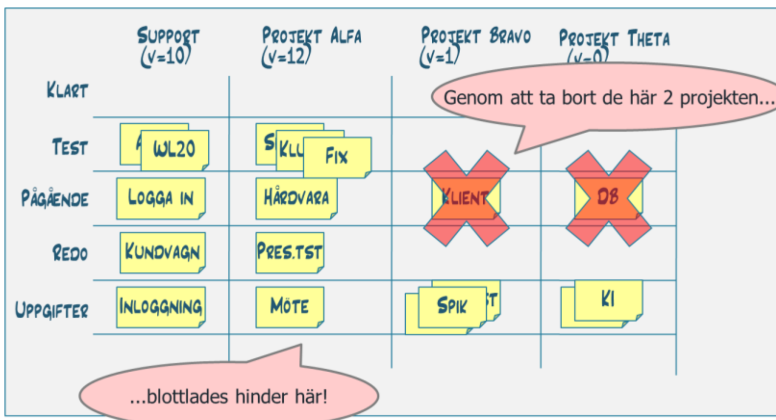
Allmänna lärdomar

När pågående arbete minskar ökar begränsningarna

Samtliga lag började med rätt generösa PA-gränser. Då lades mest energi på att försöka skapa flyt och att säkerställa att organisationen fick den support den behövde.

Till att börja med vill cheferna köra flera projekt samtidigt men inom några veckor blev det tydligt att det inte fanns tillräcklig kapacitet för att hantera de lågprioriterade projekten. Det krävdes bara en snabb blick på tavlan för att inse att inte någon av de lågprioriterade uppgifterna påbörjades. Detta föranledde cheferna att minska antalet projekt per lag.

Med tiden, när flödet för de högprioriterade uppgifterna blev stabilare, började vi dra ner PA-gränserna. Det gjorde vi genom att minska antalet pågående projekt (kolumner) från tre, till två och sedan ett. När detta skedde började begränsingar utanför laget blottläggas. Teammedlemmar började rapportera att de inte fick hjälp från utomstående i tid varpå cheferna vände sin uppmärksamhet dit för att hantera det.



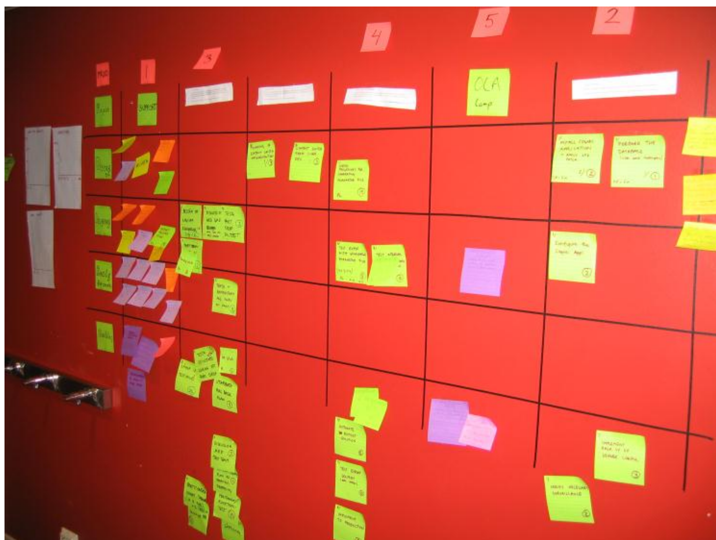
Bland de saker som uppdagades var hur tydlligt dåligt resultat från andra team påverkade vår prestation. Det var svårt att bibehålla ett mjukt och snabbt flöde när inkommande uppgifter hela tiden behövde rättas till.

De här problemen var inte helt osynliga innan vi satte igång. Det vara snarare en fråga om ”vilka problem ska vi hantera först” och att nå konsensus kring det. Genom Kanban-tavlan kunde alla se hur specifika problem påverkade *flödet* och det gjorde det lättare att samla momentum att ta itu med prolemen över organisationsgränserna.

Tavlan förändras med tiden, hugg inte utseendet i sten

Alla Kanban-tavlor förändras med tiden. Det tar vanligtvis två till tre omarbetningar innan laget hittar en som fungerar bra. Att lägga mycket tid på den första layouten är därför troligen bortkastat. Se till att du enkelt kan bygga om tavlan. Vi använde svart tejp för vår layout. Det gjorde det lätt att arrangera om tavlan och kunde användas på väggar såväl som på vita tavlor. Ett annat sätt jag har sett är att rita tavlans rutnät med en tjock märkpenna (men se till att det går att ta bort! ☺)

Nedan syns ett typiskt exempel på en optimerad layout. Prioriteterna ändrades mycket i början så istället för att behöva flytta en hel kolumn med lappar fram och tillbaka satte laget prioriteten ovanför varje kolumn.



Figur 14. Tidig Kanban-tavla med lappar för aktuell prioritet.

Var inte rädd för att experimentera och misslyckas

Lärdomen jag fick av detta äventyr var att det faktiskt inte finns någon slutpunkt. Misslyckandet sker i det ögonblick vi tror att det finns det. Det finns bara en ett ändlöst experimenterande och lärande. Att aldrig misslyckas betyder att aldrig lära sig. Vi misslyckades flera gånger på vägen (dålig tavel-layout, estimeringar, redundanta progressdiagram, etc.) men varje gång lärde vi oss någonting nytt och viktigt. Om vi hade slutat försöka, hur skulle vi då kunna lära oss?

Framgången med Kanban har nu också inspirerat både chefserna och Scrumteamen att experimentera med Kanban. Kanske kan denna bok vara till hjälp!

Några sista ord på vägen

Börja med retrospektiven!

Många möjligheter och mycket att tänka på, eller hur? Hoppas den här boken hjälpte till att skingra dimman. Det funkade i alla fall för oss :o)

Om du är intresserad av att förändra och förbättra din process, låt oss ta ett beslut åt dig redan nu. Om du inte gör retrospektiv regelbundet så börja med det! Och se till att de leder till verklig förändring. Skaffa en extern facilitator om det behövs.

När ni väl har effektiva retrospektiv på plats har ni börjat er resa i att ta fram precis den rätta processen för er situation – oavsett om den är baserad på Scrum, XP, Kanban, eller en kombination av dessa, eller något annat.

Sluta aldrig experimentera!

Kanban och Scrum är inte målet – kontinuerligt lärande är det! Snabb återkoppling är nyckeln till lärande. Så använd återkopplingen! Ifrågasätt allt, exeperimentera, misslyckas, lär och exeperimentera igen. Bekymra dig inte om att göra allt rätt från början för det kommer du ändå inte att lyckas med! Bara börja någonstans och utveckla därifrån.

Det enda *riktiga* misstaget är att inte lära från sina misstag!

Som sagt, det är där du kan lära sig mest.

Lycka till och glöm inte att ha kul på vägen!

/Henrik & Mattias, Stockholm 2009-06-24

H: Var det allt?

M: Jag tror det. Vi slutar här.

H: Vi kanske ska säga vilka vi är?

M: Bra poäng. Om vi får det att verka som att vi är ett par trevliga killar så får vi kanske konsultuppdrag.

H: Jamen, då gör vi det! Sen får det räcka.

M: Ja, vi har annat att göra och det har läsarna också.

H: Faktum är att min semester börjar nu :o)

M: Hörru, var lagom kaxig.

Om författarna

Henrik Kniberg och Mattias Skarin är konsulter på Crisp i Stockholm. De tycker om att hjälpa företag att lyckas med både den tekniska och mänskliga sidan av programvaruutveckling och de har hjälpt dussintals företag att omsätta Lean och Agila principer i praktiken.

Henrik Kniberg



Det senaste årtiondet har Henrik varit utvecklingschef på 3 svenska IT-företag och hjälp många fler att förbättra sina processer. Han är certifierad Scrum-utbildare och arbetar regelbundet med pionjärer inom Lean och Agile som Jeff Sutherland, Mary Poppendieck och David Anderson.

Henriks förra bok, “Scrum and XP from the Trenches” har mer än 150,000 läsare och är en av de mest populära böckerna i ämnet. Han har fått pris för bästa talare flera gånger för presentationer på internationella konferenser.

Henrik växte upp i Tokyo och bor nu i Stockholm med sin fru Sophia och tre barn. På fritiden är han aktiv musiker och komponerar musik och spelar bas och keyboard i lokala band.

henrik.kniberg@crisp.se

http://blog.crisp.se/henrikkniberg

http://www.crisp.se/konsulter/henrik.kniberg

Mattias Skarin



Mattias arbetar som Lean-coach och hjälper programvaru-företag dra nytta av Lean och Agile. Han handleder på alla nivåer, från utvecklare till ledning. Han har hjälp ett spelföretag att minska utvecklingstiden från 24 till 4 månader, återställt förtroendet hos en hel utvecklings-avdelning och var en av de första pionjärerna i Kanban.

Som entreprenör har han varit med och grundat och drivit två företag.

Mattias har en civilingenjörsexamen i kvalitetsledning och har arbetat som utvecklare av verksamhetskritiska system i 10 år.

Han bor i Stockholm, gillar rock ’n’ roll, dans, racing och skidåkning.

mattias.skarin@crisp.se

http://blog.crisp.se/mattiasskarin

http://www.crisp.se/konsulter/mattias.skarin